République algérienne démocratique et populaire

Ministère de l'enseignement supérieur et de la recherche scientifique
Université SAAD DAHLEB –BLIDA -

Mémoire de Fin D'étude

Pour l'obtention de diplôme de master en informatique

Option : Ingénierie du logiciel

Thème : Architecture a base d'agents mobiles sécurisés pour la recherche d'information

Réalisé par : KOBBI IMENE

Encadré par : Mr. DJAMEL EDDINE MENACER

Mme Aoussat.
Mme Fatah.
Mme Bouneb ah.

Promotion : 2008/2009

Dédicaces

Je dédie ce travail

A la mémoire de mon père « Mouhamed »
A mon beau père et a ma très chère maman qui ma toujours soutenue et encouragée et crue en moi à tout moment, que Dieu la garde en bonne santé toujours a mes cotés.
A ami HAMID qui ma soutenu et été pour moi un exemple à suivre ainsi qu'un parent et aussi a ma chère Tata FATIHA
A mon grand-père et ma grand-mère.
A mon cher mari HAMZA qui a été toujours présent surtout dans mes moments difficile en me donnent réconfort et aide.
A mes chères tantes surtout «HASSINA, SOUAD et MAHIA leurs maris Didou ALLAL et Didou TOUFIK
A mes frères ALI, OUSSAMA, TADJOU et l'adorable AMIROU.
A mes meilleurs amis et sœur : SARAH, IMEN, HADJIRA, WAHIBA, RACHIDA.
A ma très cher petite sœur KSOURI MARIA
A tous ceux qui mon aidés de loin ou de prés.

IMENE .k

Sommaire :

Comparaison

Peer To Peer

Chapitre 3:

La Recherche d'information

Partie II :

Chapitre 4: Analyse et conception

I. Description du système :

II. Conception du système

Liste Des Figures

Partie 1 :

Chapitre 1 : client /serveur

Chapitre 2 : La mobilité

Partie III:

Chapitre5 : Implémentation

Partie IV:

Chapitre6 : Tests

Introduction générale

Contexte :

L'émergence des d'applications de plus en plus complexes telles les applications multimédias et la recherche d'information a connu un développement exponentiel avec le développement des applications réparties qui ont étés nés avec internet, le protocole client-serveur qui a semblé jusqu'ici répondre au besoin des utilisateurs car cette technologie consiste a migré un processus en exécution d'une machine a une autre afin de bénéficier des de leur puissance de calcule. Mais la saturation du réseau est le majeur problème, pour cela une nouvelle approche vient défier cette dernière et le complémenter, c'est l'approche agents mobiles.

Problématique:

Les agents mobiles qui eux sont des programmes exécutables qui se déplacent d'une machine source vers une machine destination pour s'y exécuter. Ils ont comme principales caractéristiques l'autonomie, l'intelligence et surtout la mobilité. Et il court un très grand point faible c'est la sécurité qui est un principale point de faille de cette technologie.

Plusieurs solution sont ont étés proposés pour régler le problème de sécurité des agents tel que la signature numérique des agents mais ces solutions dégradent les performances des agents.

Objectif :

Notre objectif est réalisation d'un site de recherche d'information, il s'agit d'une application basée sur les agents mobiles à travers l'architecture proposée IMAN (Interaction Mobile Agent in Nail) se dernier va essayer au mieux de comblé la non sécurité des agents mobiles.

Notre travail consiste à concevoir une architecture à base agents mobiles a travers un prototype proposée (modèle buyer-seller) afin de le rendre compatible aux applications de recherche d'information puis faire la comparaison avec d'autre modèles afin d'extraire leur niveau de performance.

Organisation du mémoire:

Notre travail consiste à comprendre le fonctionnement de deux architectures à savoir le paradigme client-serveur et celui des agents mobiles ainsi que le concept du recherche d'information. Pour cela, nous avons entrepris notre étude selon les 6 chapitres suivants :

Chapitre I : est consacré à une étude sur le modèle client-serveur (présentation, implémentations, avantages et limites…)

Chapitre II : vu que nous avons utilisé des agents mobiles, il s'avère indispensable de faire une étude générale sur le paradigme d'agent et les systèmes multi-agents (définitions, architecture interne, propriétés,…) puis une étude approfondie sur les agents mobiles.

Chapitre III : Ce chapitre est consacré à une étude globale sur le concept de recherche d'information

Chapitre IV : Pour pallier aux limites des architectures traditionnelles Client/serveur présentés au chapitre I et pour essayer de contribuer à la résolution des problèmes liés à l'utilisation des agents mobiles dans les applications RI déjà vus dans les chapitres II et III , nous présentons dans ce chapitre l'architecture IMAN(Interaction Mobile Agent in Nail) permettant d'utiliser pleinement les agents mobiles de manière sure et efficace.

Chapitre V : est consacré à la conception implémentation de notre système à travers l'architecture IMAN décrite dans le chapitre IV.

Chapitre VI : dans se chapitre on va faire des teste sur notre système et comparé ces résultat avec un autre prototype afin d'en jugé des apports de notre système dans le domaine de la recherche d'information par agents mobiles

Enfin, nous terminons par une conclusion générale, qui résume l'apport essentiel de notre travail, qui tente de s'ouvrir sur de nouveaux éléments de réflexions et propose quelques perspectives de recherche. Deux annexes sont ajoutées à la fin pour éclaircir les notions non approfondies dans les chapitres.

Partie I :

Etat de L'art

Chapitre 1 :

Les systèmes distribués : Technologie client serveur

1- Introduction :

Les systèmes répartis se compose d'une collection d'ordinateurs autonomes joints par un réseau informatique et équipés d'un logiciel réparti qui leurs permets de coordonner leurs activités et de partager les ressources du système: équipement, logiciel et données, et cela invoque l'utilisation de différents modèles de communication pour le développement d'applications reparties .Dont :

- Le modèle a événement asynchrone
- Le modèle a mémoire virtuelle partagée
- Modèle basé sur le code mobile
- Modèle clients serveurs, dont il va faire l'objet de se chapitre

2-Le modèle client serveur:

Les systèmes informatiques peuvent être classés en deux grandes catégories :
Les systèmes centralisés reposant sur des mainframes et les systèmes distribués qui sont construits selon deux modèles : le modèle pair à pair qui peut être pur ou hybride et le modèle client/serveur plat ou hiérarchique :

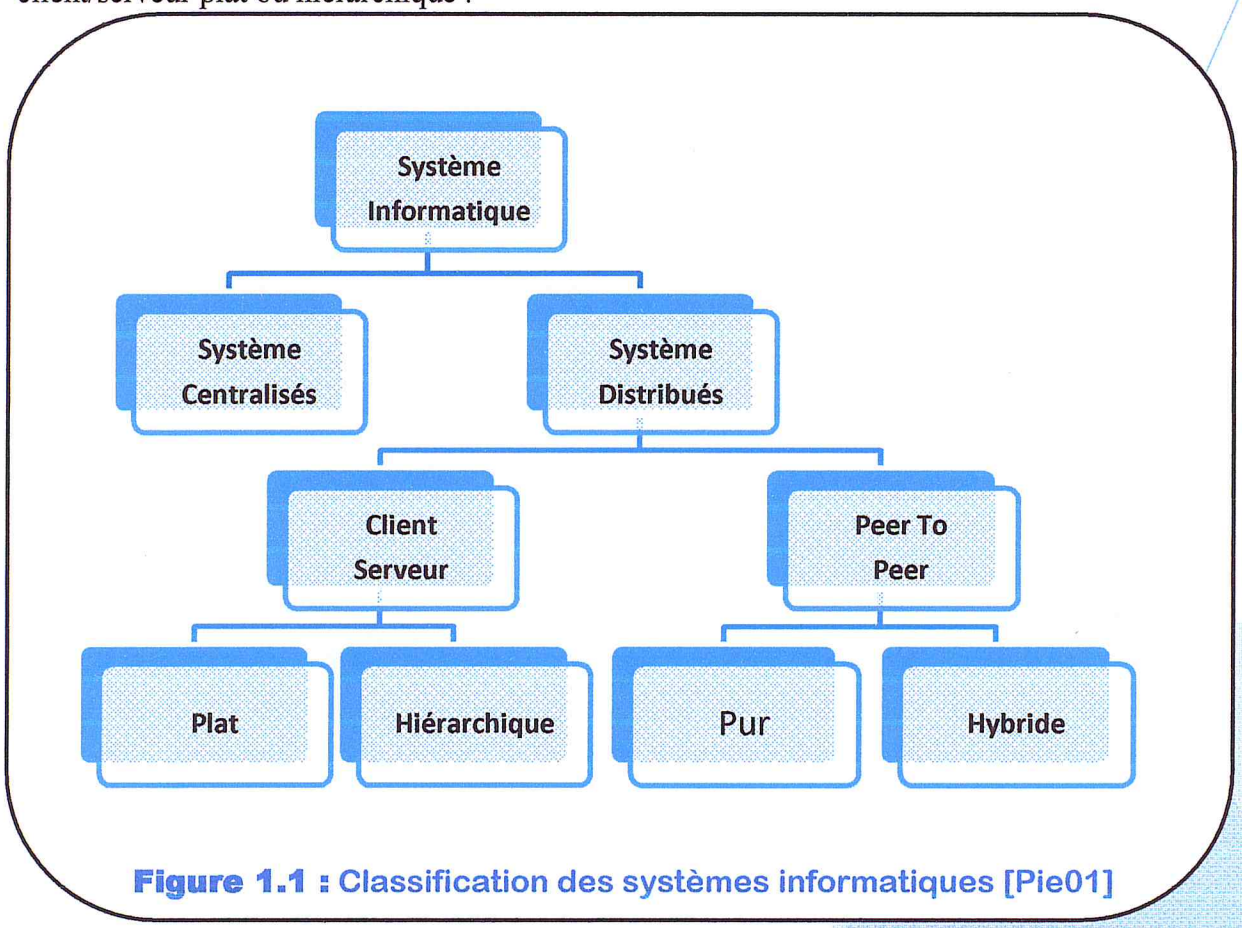


Figure 1.1 : Classification des systèmes informatiques [Pie01]

3. Composent de base du Modèle Client Serveur :

« En fait, l'architecture client serveur est plus large. Un réseau n'est pas toujours nécessaire. Il est possible de la réaliser sur une même machine (et pour quoi pas un *mainframe !)* en dégageant deux processus, l'un-**le client-** qui envoie des **requêtes** à l'autre – **le serveur-**, ce dernier traitant les requêtes et renvoyant des **repenses**.» **[Pie01]**

- **Client** : processus demandant l'exécution d'une opération à un autre processus par envoi de message contenant le descriptif de l'opération à exécuter et attendant la réponse par un message en retour.
- **Serveur** : processus accomplissant une opération sur demande d'un client, et lui transmettant le résultat par une repense.
- **Requête** : message transmis par un client à un serveur décrivant l'opération à exécuter pour le compte du client.
- **Réponse** : message transmis par un serveur à un client suite à l'exécution d'une opération, contenant les paramètres de retour de l'opération. Ses Notion peuvent êtres schématisé sur la figure suivante :

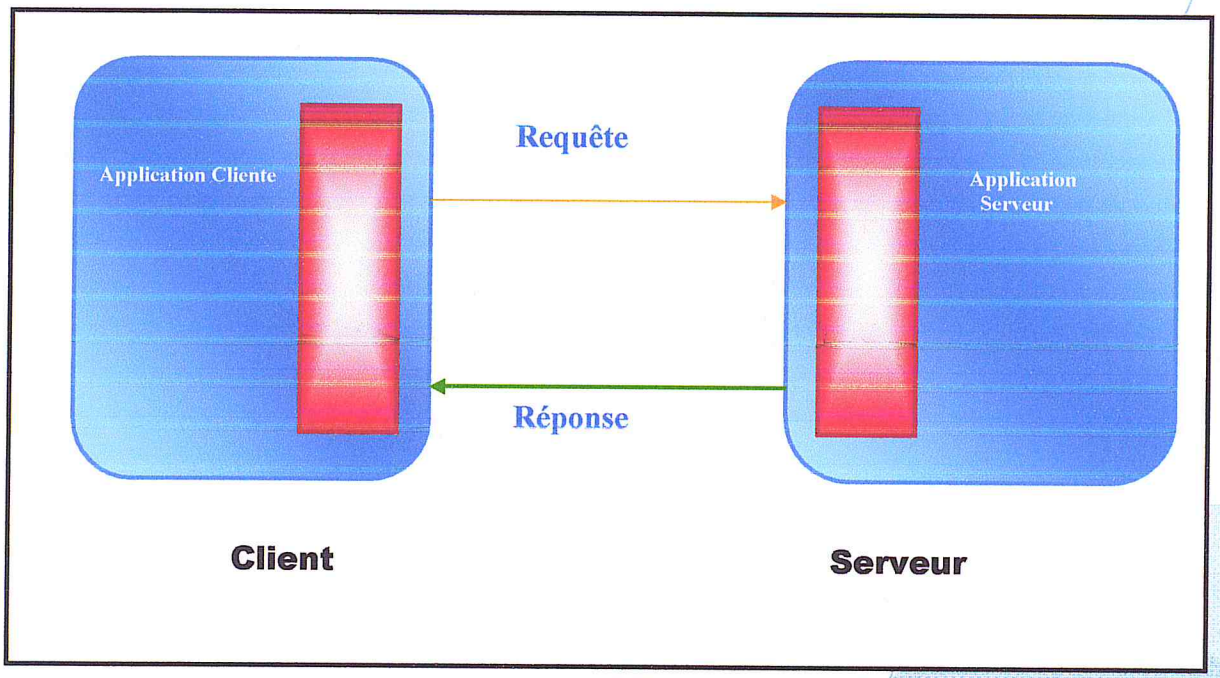


Figure 1.2: Le modèle de base Client Serveur

4- Techniques de dialogue :

Le client a besoin d'assurer le contacte avec le serveur afin d'envoyer la requête et recevoir la réponse pour cela plusieurs protocoles existes sur une couche système qui localise les sites le transport et la communication entre ces sites distant.

4.1-Message de dialogue :

Quand le client exécute une application et demande l'exécution d'une opération. Les quatre services de transport seront activés :

- **SendRequest () :** transmis le message décrivant la requête à une adresse correspondant à la porte d'écoute du serveur.
- **ReceiveReply () :** permet au client de recevoir la repense envoyée par le serveur.
- **ReceiveRequest () :** permet au serveur de recevoir la requête du client sur sa porte d'écoute.
- **SendReply () :** permet au serveur d'envoyer la repense sur la porte d'écoute du client.

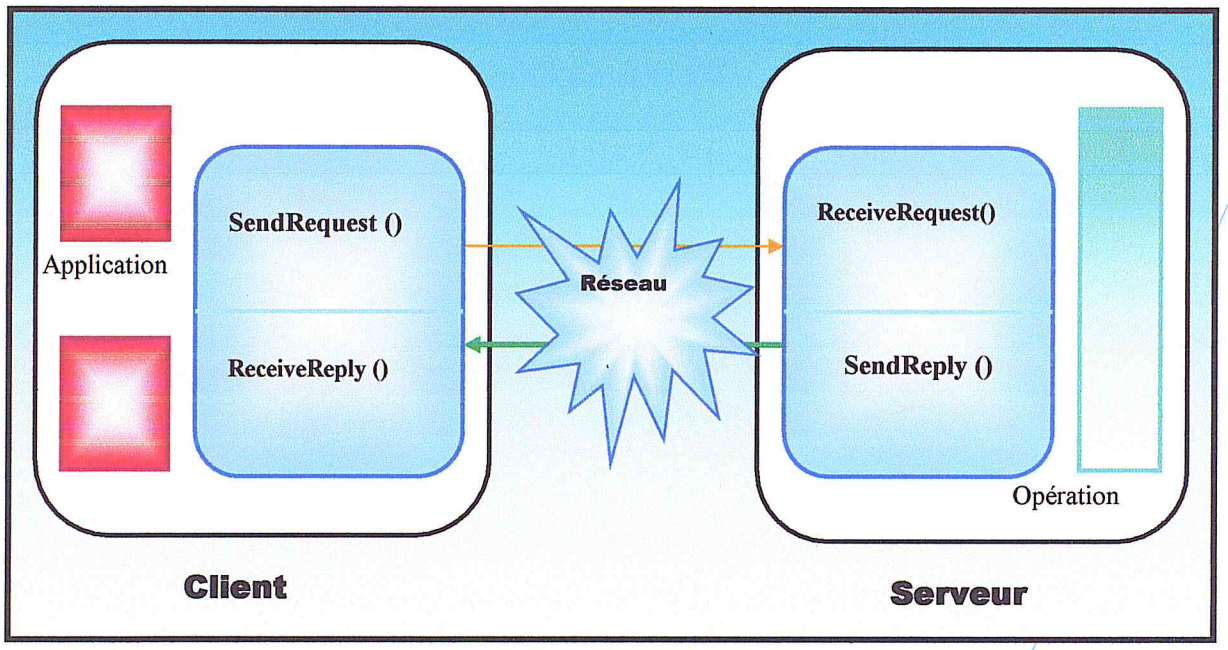


Figure 1.3: Dialogue entre client serveur

4.2-Appel de procédure à distance RPC :

« *RPC (Remote Procedure Call) :* Est une technique d'appeler une procédure distante comme une procédure locale, en rendant transparent les messages échangés et les assemblages/désassemblages de paramètres. » [Pie01]

Le RPC offre les services de l'appelé à l'appelant sur le site de ce dernier. Il est réalisé par introduction d'une ***souche***[1] ***(stub)*** pour transformer l'appel de procédure en un envoie de message depuis le site de l'appelant au site de l'appelé.

Une souche de procédure symétrique reçoit le message et réalise l'appel effectif de l'appelé par débranchement. Les paramètres de retour transitent par un chemin inverse via le réseau d'interconnexion.

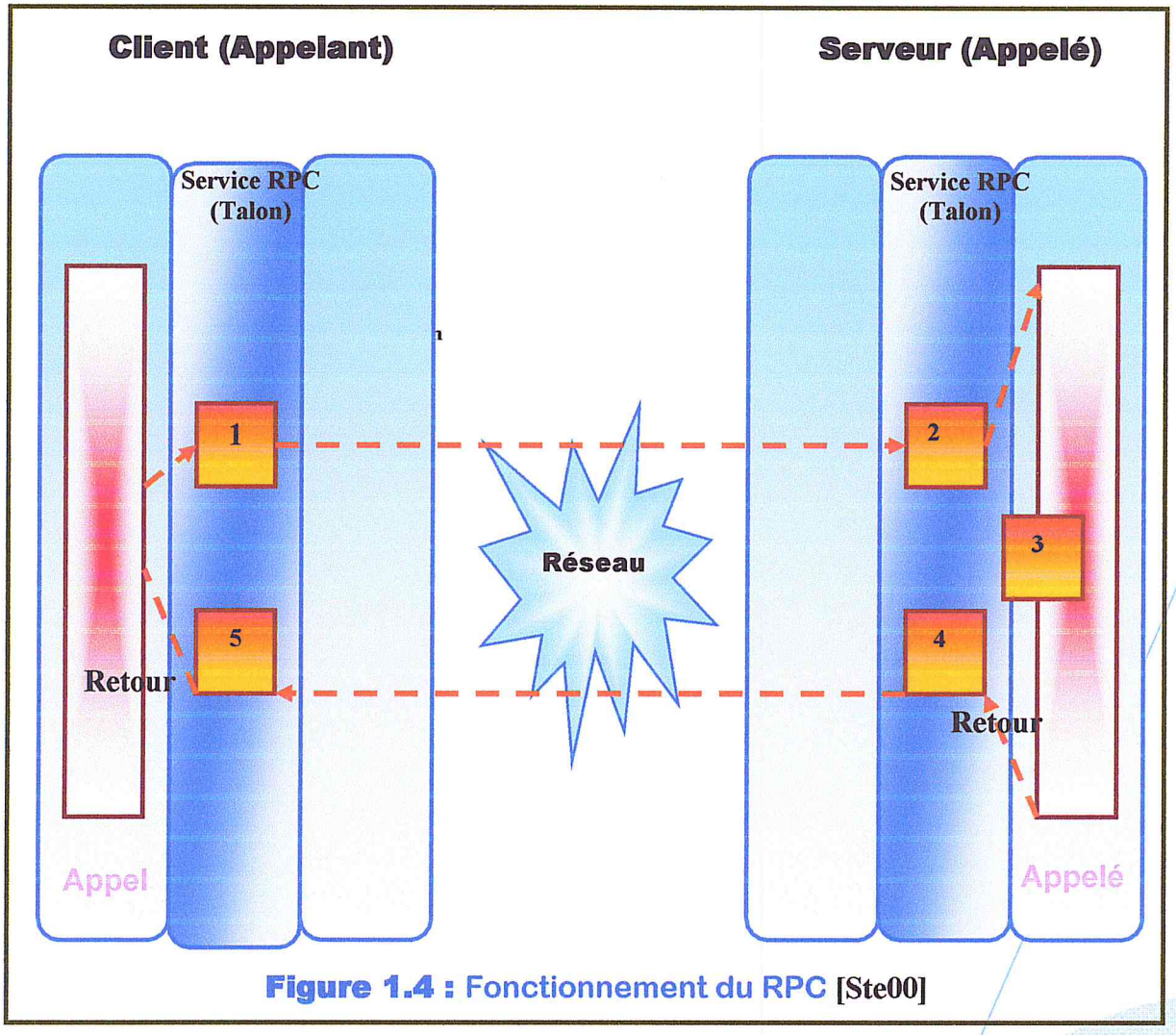


Figure 1.4 : Fonctionnement du RPC [Ste00]

[1] Représentant d'une procédure sur un site client ou serveur capable de recevoir un appel de procédure du client et de le transmettre en format adapté à l'implémentation ou à son représentant

Talon client (*stub*): C'est la procédure d'interface du site client

- qui reçoit l'appel en mode local
- le transforme en appel distant en envoyant un message
- reçoit les résultats après l'exécution
- retourne les paramètres résultats comme dans un retour de procédure

Talon serveur (*skeleton*) : C'est la procédure sur le site serveur

- qui reçoit l'appel sous forme de message
- fait réaliser l'exécution sur le site serveur par la procédure serveur (choix de la procédure)
- retransmet les résultats par message

4.3/ Appel de méthode à distance :

Un objet est une entité logicielle regroupent en son sein des données et un comportement chaque objet réalise une tache précise et il est accessible a travers une interface qui comporte un ensemble de méthode, une application autant été décrite comme un ensemble d'objet et d'interaction entre objet

5. Client serveur Orienté Objet :

L'Object Management Group (OMG) est un consortium international créé en 1989 des producteurs de logiciel (Netscape), L'objectif de ce groupe est de faire émerger des standards pour l'intégration d'applications distribuées hétérogènes à partir des technologies orientées objet. Afin d'assurer la réutilisation, l'interopérabilité et la portabilité de composants logiciels.

L'élément clé de la vision de l'OMG est **CORBA (Common Object Request Broker Architecture)**: un « middleware » orienté objet. Ce bus d'objets répartis offre un support d'exécution masquant les couches techniques d'un système réparti (système d'exploitation, processeur et réseau) et il prend en charge les communications entre les composants logiciels formant les applications réparties hétérogènes. **[Lio04]**

5.1/ Le modèle CORBA:

5.1.1/ Architecture CORBA :

L'OMG définit aussi une vision globale de la construction d'applications réparties : l'Object Management Architecture Guide [OMAG 95]. Cette architecture globale, appelée aussi l'OMA, vise à classifier les différents objets qui interviennent dans une application en fonction de leurs rôles :

Architecture CORBA se base sur plusieurs composants :

1. **ORB (Object Request Broker)** :

Gère les transferts entre les clients et les serveurs c'est le noyau de transport des requêtes aux objets. Il intègre au minimum les protocoles GIOP et IIOP. L'interface au bus fournit les primitives de base comme l'initialisation de l'ORB

2. **IDL (langage de définition d'interfaces)** :C'est un langage de définition des services proposés par un objet

- Une interface comprend un ensemble d'opérations nommées et des paramètres sur ces opérations
- L'IDL est le moyen par lequel un objet particulier indique à ses clients potentiels les opérations disponibles et la façon des les invoquer
- C'est un langage neutre et totalement déclaratif
- A partir des définitions d'IDL, il est possible de mettre en correspondance (mapping) les objets CORBA dans un langage de programmation particulier ou avec des objets systèmes.
- Il définit les types d'objet par spécification des interfaces

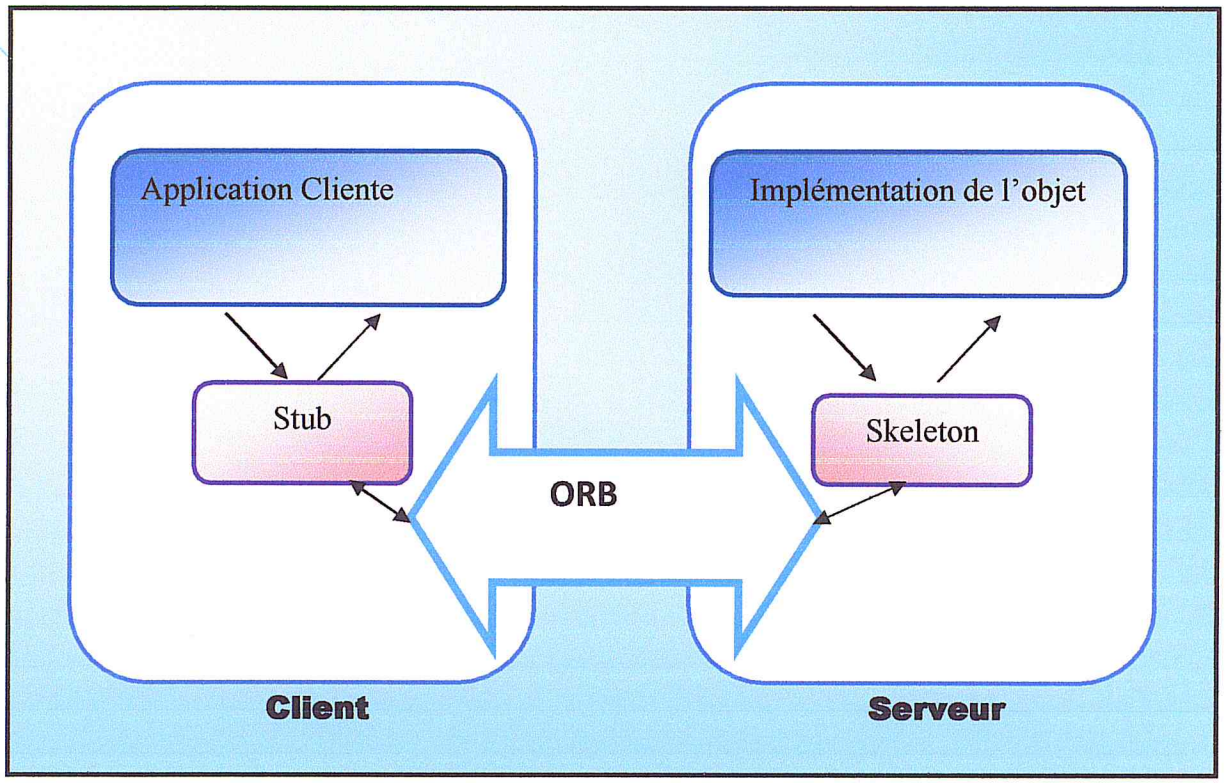


Figure 1.5 : Dialogue entre Client ORB Serveur

3. Basic Object Adapter (BOA): est l'adaptateur d'objets qui s'occupe de créer les objets CORBA, de maintenir les associations entre objets CORBA et implantations et de réaliser l'activation automatique si nécessaire.

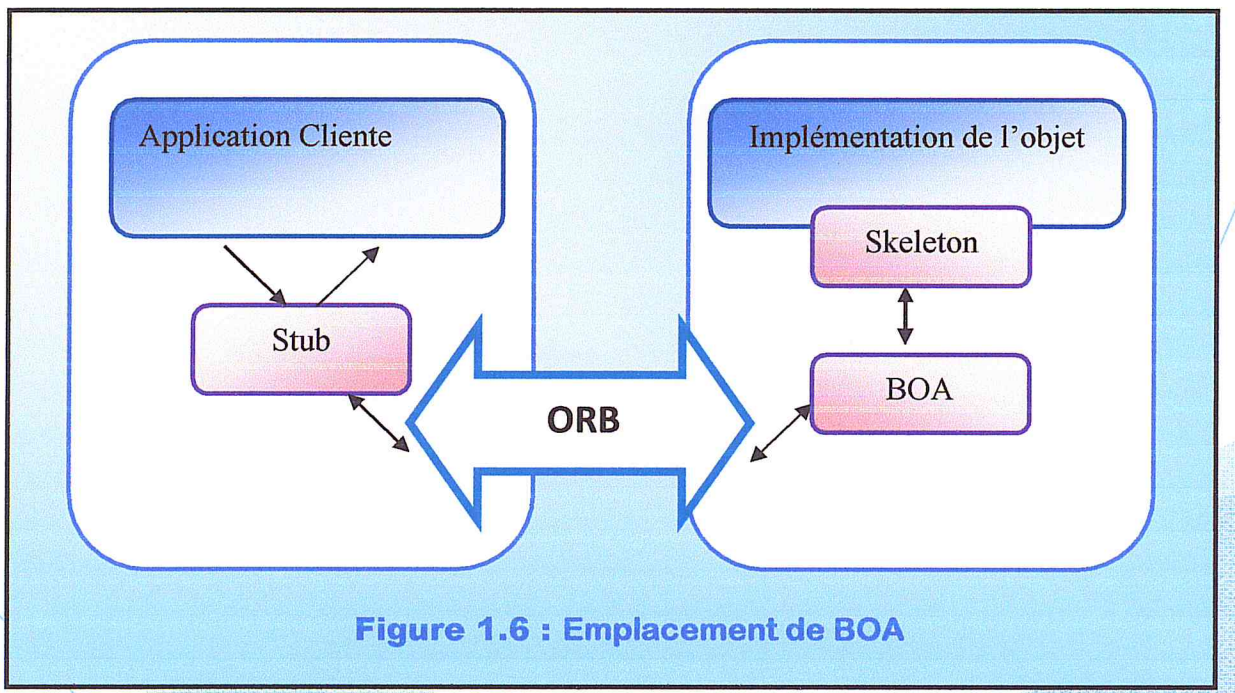


Figure 1.6 : Emplacement de BOA

4. **IIOP** (Internet Inter ORB Protocol) :C'est le protocole de communication utilisé qui présente une spécialisation du protocole GIOP (Général Inter ORB Protocol) adapté au TCP/IP.
5. **DSI** *(Dynamic Skeleton Interface)* est l'interface de squelettes dynamiques qui permet d'intercepter dynamiquement toute requête sans générer une interface SSI. C'est le pendant de DII pour un serveur.
6. **DII :** *(Dynamic Invocation Interface)* est l'interface d'invocations dynamiques permettant de construire dynamiquement des requêtes vers n'importe quel objet CORBA sans générer/utiliser une interface SII.

7. **IFR :** *(Interface Repository)* est le référentiel des interfaces contenant une représentation des interfaces OMGIDL accessible par les applications durant l'exécution.
8. **ImplR** *(Implementation Repository)* est le référentiel des implantations qui contient l'information nécessaire à l'activation. Ce référentiel est spécifique à chaque produit CORBA.

5.1.2/ Fonctionnement de CORBA: [Lio04]

Il est fondé sur une entité virtuelle, l'objet CORBA, gérée par le bus CORBA. Chaque application peut exporter ses services sous la forme d'objets CORBA. La coopération client serveur se déroule de la manière suivante :

1. Le client détient une référence sur un objet CORBA qui permet de le localiser sur le bus.
2. Le client dispose de l'interface de l'objet CORBA (type abstrait de l'objet CORBA) qui définit ses opérations et ses attributs (exprimés dans le langage IDL).
3. Le client réalise une requête (invoque) une opération sur l'objet CORBA.
4. Le bus CORBA achemine cette requête vers l'objet CORBA tout en masquant les problèmes d'hétérogénéité liés aux langages, systèmes d'exploitation, machines, réseaux.
5. L'objet CORBA est associé à un objet d'implantation
6. Le serveur détient l'objet d'implantation qui code l'objet CORBA (cette implantation pouvant évoluer au cours du temps) et gère son état temporaire.

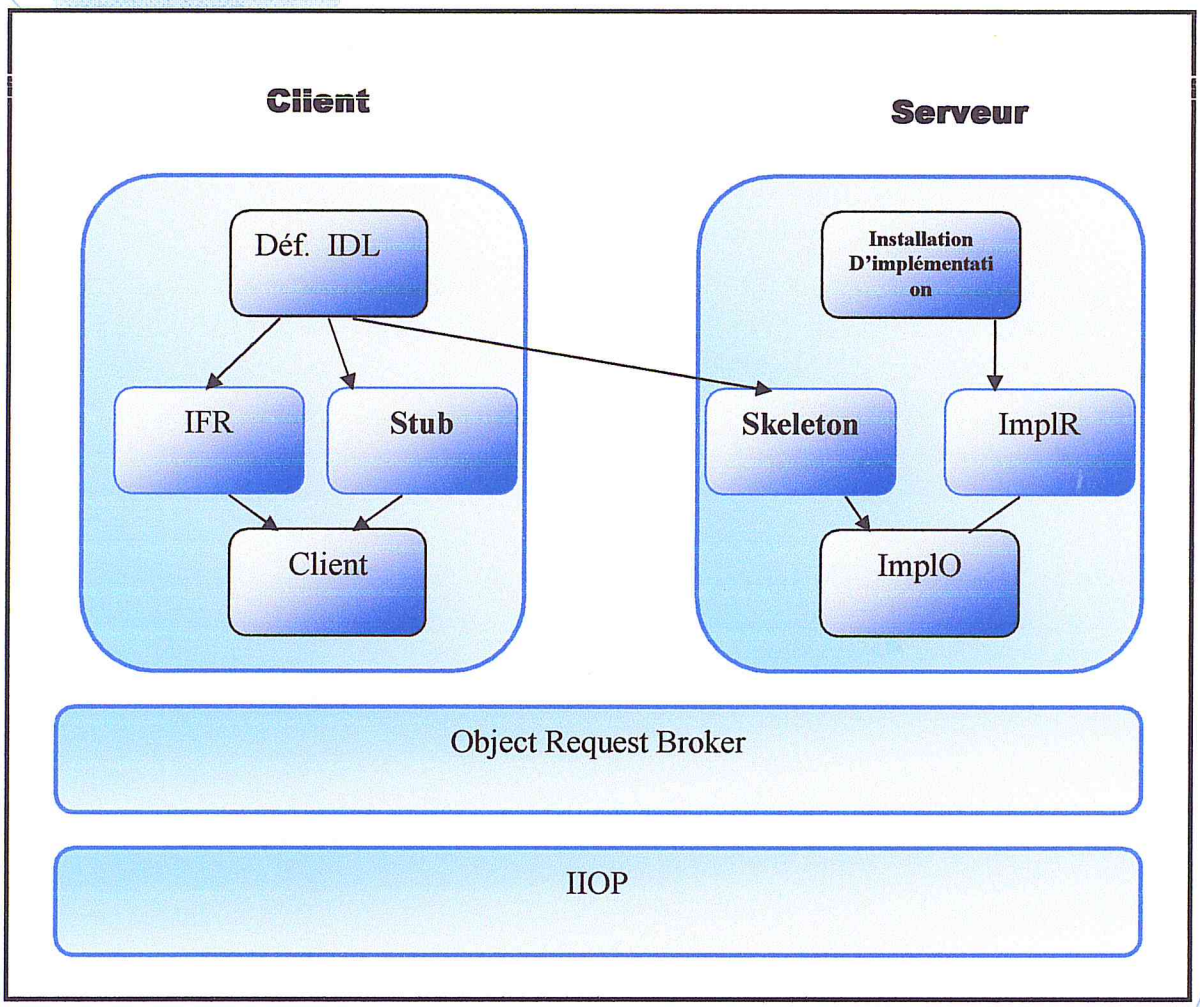


Figure 1.7: Fonctionnement de CORBA [Lio04]

5.2/ Modèle RMI :

5.2.1/Définition :

RMI (Remote Method Invocation) c'est une API java permettent de manipuler des objets distants de manière transparente pour l'utilisateur

5.2.2/ Structure des couches RMI :

En RMI la transmission des données se fait a travers un système de couches basés sur le modèle OSI afin de garantir une interopérabilité quand aux connexions ils sont faits grâce au protocole JRMP (Jva Remote Method Protocol) basé sur TCP/IP.

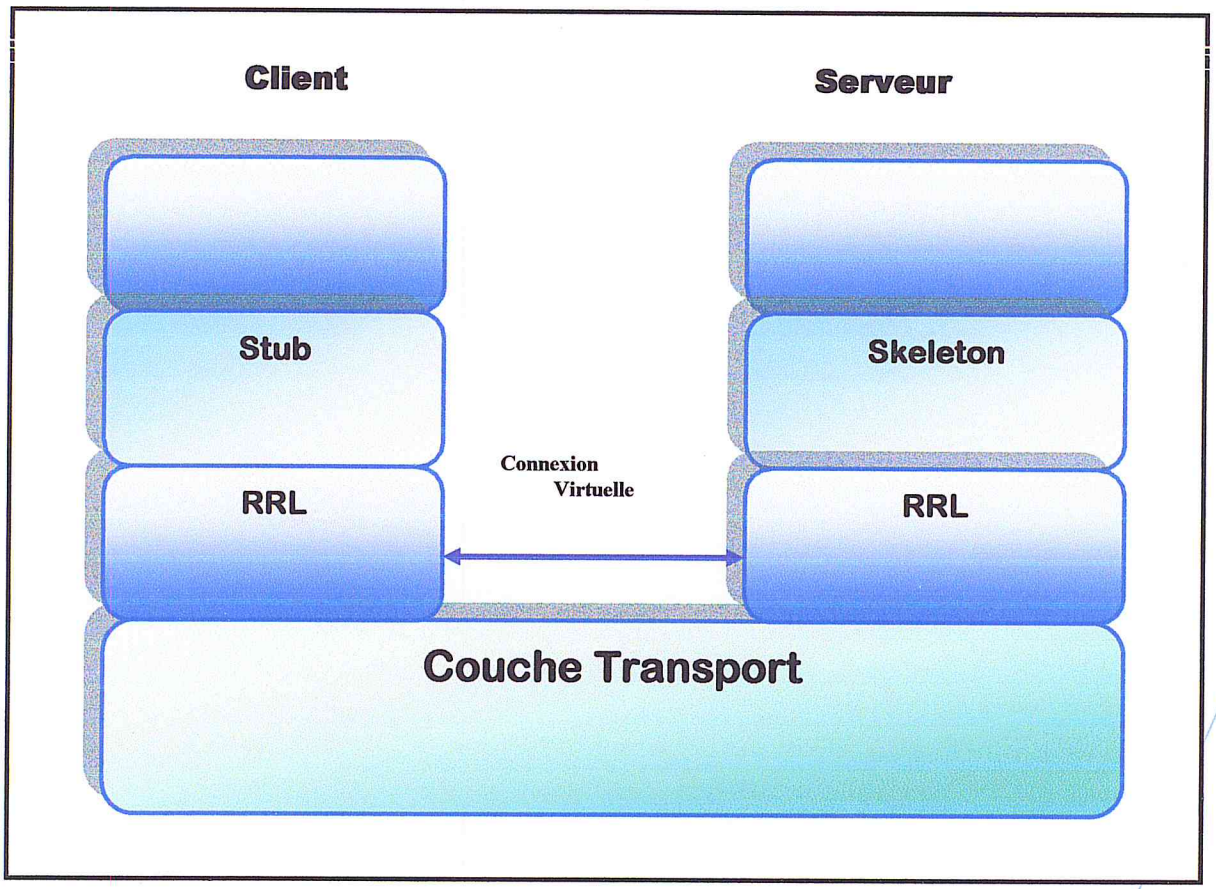


Figure 1. 8: Structure des couches RMI

- ***Le stub et le skeleton :*** respectivement sur le client et serveur Assure la conversion des communications avec l'objet distant
- ***La couche de référence*** *(RRL, Remot Reference Layer)* **:** Charger du système de localisation afin d'assurer un moyen au objet d'avoir des références a l'objet distant appeler « ANNUAIRE RMI »
- ***La couche de transport :*** Permet d'écouter les appelles entrent ainsi que d'établir les connexions et le transport des données sur le réseau

5.2.3/ Architecture RMI :

Lorsqu'un client veut invoquer une méthode d'un objet distant, il effectue les opérations suivantes :

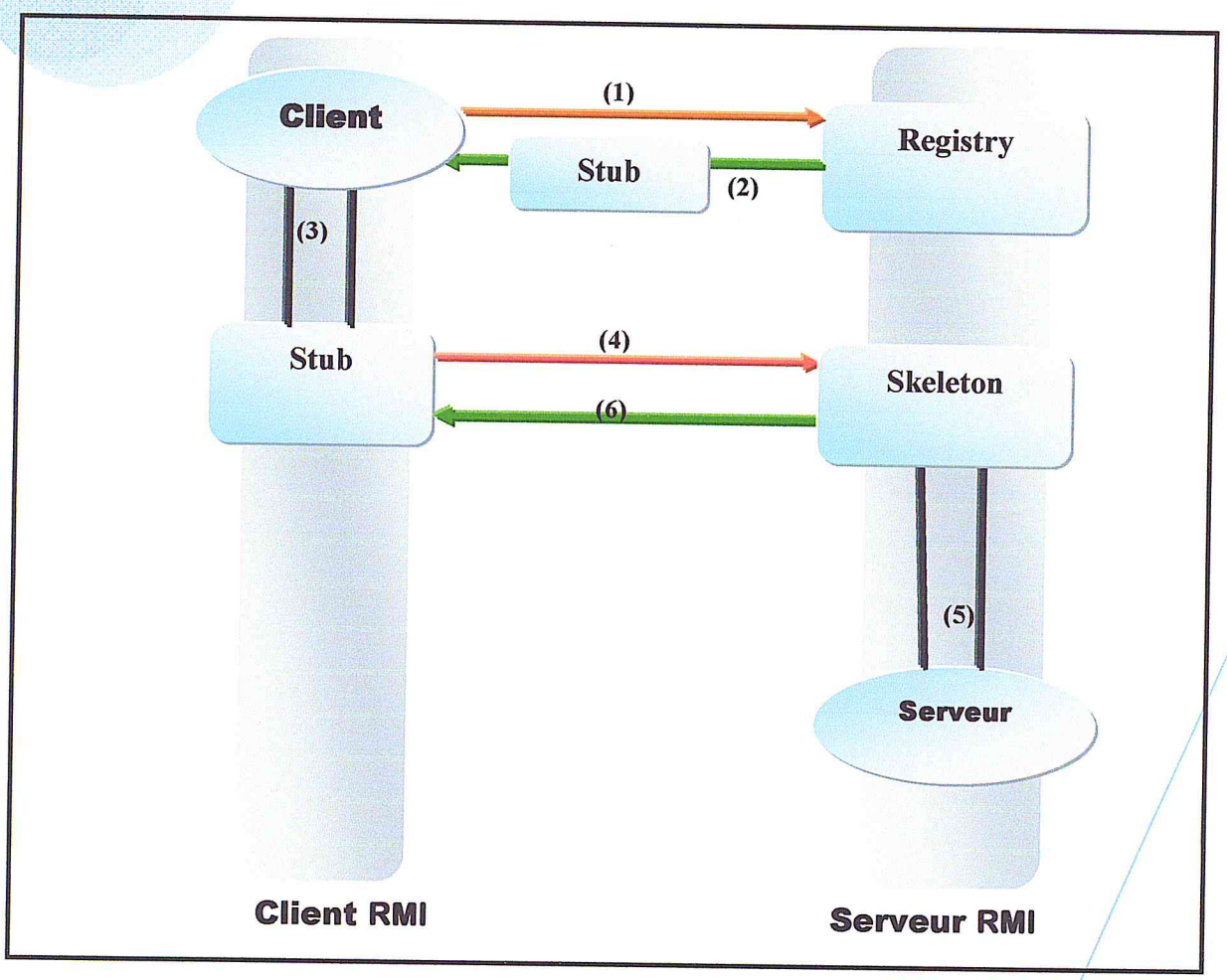


Figure 1.9 : Architecture RMI

1. il localise l'objet distant grâce à un service de désignation : le **registre RMI**
2. il obtient dynamiquement une image virtuelle de l'objet distant (**stub**). Le stub possède exactement la même interface que l'objet distant.
3. Le stub transforme l'appel de la méthode distante en une suite d'octets, c'est ce que l'on appelle la sérialisation, puis les transmet au serveur en instanciant l'objet sous forme de flot de données. On dit que le stub *"marshalise"* les arguments de la méthode distante.
4. Le **squelette** instancié sur le serveur "désérialise" les données envoyées par le stub (on dit qu'il les "*démarshalise*"), puis appelle la méthode en local
5. Le squelette récupère les données renvoyées par la méthode (type de base, objet ou exception) puis les marshalise
6. le stub démarshalise les données provenant du squelette et les transmet à l'objet faisant l'appel de méthode à distance

5.3. .Net Remoting:

5.3.1. Présentation de .Net Remoting :

Dot Net Remoting est une solution MICROSOFT pour interopérabilité client/serveur entre objets .Net distants, il est le successeur de DCOM et très proche de JAVA RMI et de CORBA/IIOP. (Internet Inter-ORB-Protocol)

En d'autre terme, dot Net Remoting est un moyen de faire dialoguer de façon dynamique des applications séparées par la barrière de leur processus ou par la distance entre les machines qui les font tourner.

Le Remoting sous .Net consiste en un ensemble de services qui permettent d'accéder par un réseau à des objets existants en local sur la même machine, ou sur le réseau.

5.3.2. Principe du .Net Remoting

Le principe général de Dot Net Remoting est d'exposer sur une partie serveur des objets accessibles par des clients distants. Les clients et les serveurs doivent tourner sur la plateforme .Net.

Les objets ou leurs références pourront donc transiter via le réseau grâce au principe de Marshalling. On peut résumer le Remoting de .Net à ceci :

A chaque fois qu'un objet se situant dans un AppDomain[2] doit communiquer avec un autre objet se trouvant dans un AppDomain[2] différent il faut utiliser .Net Remoting.

Les protocoles et formatages possibles sont paramétrables : TCP+Binaire ou http+SOAP par exemple. Par défaut, le framework .Net propose TCP et http, mais la technologie .Net permet des éditeurs tiers de réaliser leurs propres implémentations et rend donc .Net Remoting potentiellement ouvert.

[2] Les boîtes dans les quelles s'exécutent les applications .Net.

5.3.3. Fonctionnement du .Net Remoting

La communication entre deux applications .Net peut être schématisée par la figure ci-dessous :

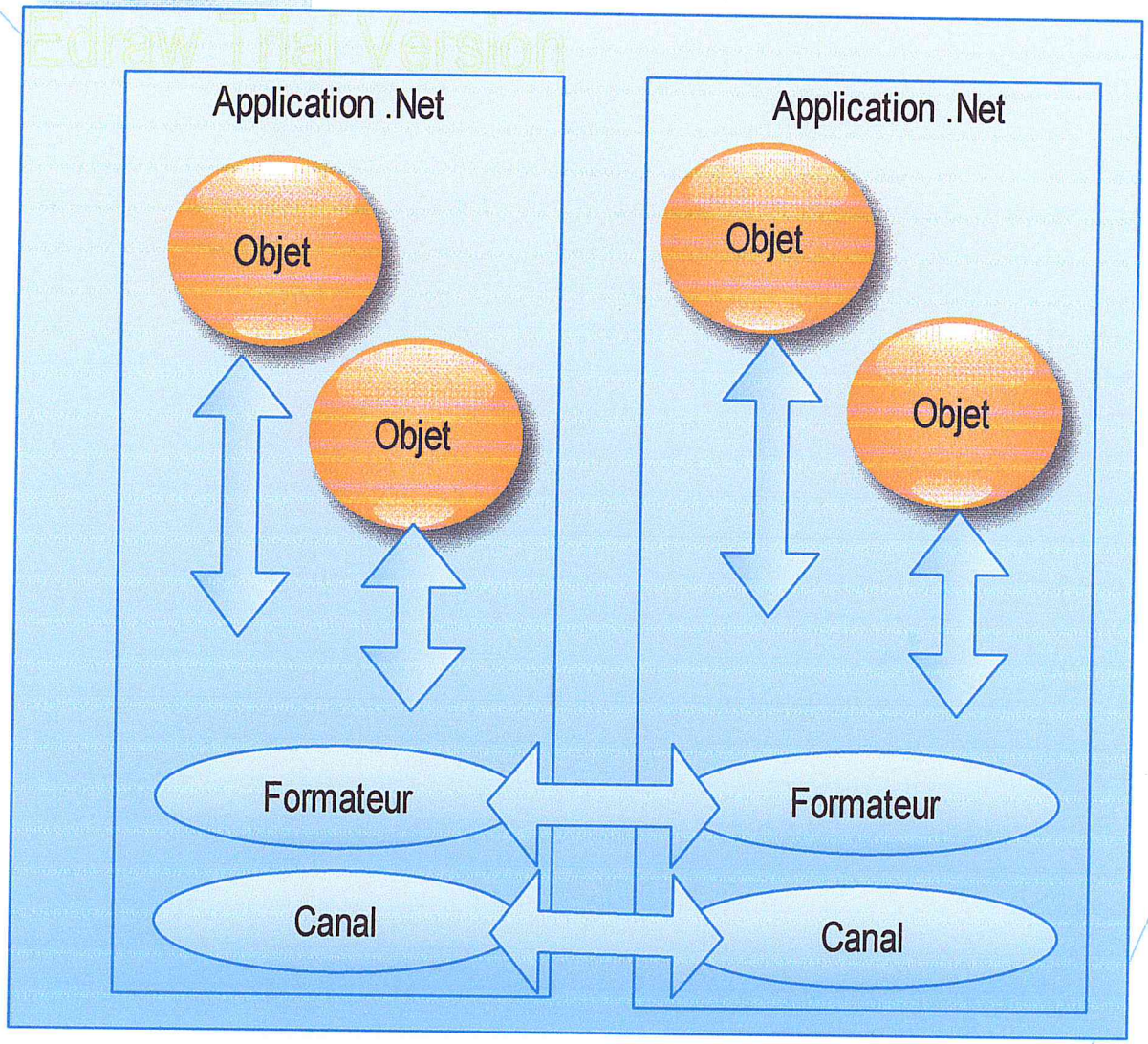


Figure1.10 : Communication entre deux applications .Net

Une application .Net contient des objets ces derniers passant par un formateur pour être ensuite transmises à l'autre application via un canal:

1. Les formateurs :

Les formateurs sont des objets capables de transformer les informations à transporter dans un format donné qui doit être connu des participants à la communication.

Un formateur a la charge de l'encodage des messages qui sont envoyés mais aussi du décodage de ceux qui sont reçus. Sur DotNet on détient les deux formateurs:

1). System.Runtime.Serialization.Formatters.Soap.SoapFormatter:

Code et décode les messages en suivant le formalisme SOAP, on l'utilise lorsqu'on choisit le protocole http.

2). System.Runtime.Serialization.Formatters.Binary.BinaryFormatter :

Utilise un mode totalement binaire et les messages peuvent être envoyés via http, TCP ou autres.

2. Les canaux :

Une fois que les messages sont encodés par un formateur, ils sont transmis via les canaux, ces derniers transmettent les messages formatés dans un sens ou dans l'autre. Un canal est un câble virtuel tiré entre deux applications qui désirent s'échanger des messages donc devant se mettre d'accord sur le canal à utiliser, à l'heure actuelle le framework .Net propose deux canaux spécialisés :

1). System.Runtime.Remoting.Channels.Http.HttpChannel :
Permet une communication via http

2). System.Runtime.Remoting.Channels.Tcp.TcpChannel:
Permet une communication via TCP

3. Les objects :

Matière première de toute application. Avant même que les messages ne soient mis en forme par un formateur, il est nécessaire que l'objet soit marshalé.
.Net propose deux types de marshaling :

1). Marshal by value (MBV) : les objets sont totalement sérialisés et transmis au client qui recréé une copie de l'objet original, les méthodes de l'objet qui sont alors invoquées sont exécutées par la copie locale qui vient d'être clonée.

2). Marshal by reference (MBR) : l'objet considéré reste sur le serveur plutôt que de véhiculer un clone de l'objet ce sont les appels à ses méthodes qui sont transmises.

5.3.4. Quelques avantages de .Net Remoting

.Net Remoting a été conçu dans l'optique informatique intégré et « agile », cette agilité n'est pas seulement un concept commercial, Microsoft a réellement tenté de simplifier les choses même pour les développeurs, les principaux atouts de cet outil sont :

- Simplicité de développement
- Simplicité du déploiement
- Simplicité de la gestion des versions

COMPARAISON

Introduction :

Après cette petite étude concernant les différentes technologies et applications pour la réalisation des systèmes distribués, donc nous avons besoins d'une comparaison entre ces technologies afin de tirer les avantages et les inconvénients de chaque architecture selon les contraintes de :

1-La Performance :

RMI est interpréteur de *Bytecode* par rapport au CORBA et Dotnet qui sont compilés donc RMI est moins performant que CORBA et Dotnet c'est-à-dire il est plus lent.

2- L'Utilisation :

Pour CORBA et Dotnet L'utilisation est difficile pour un débutant contrairement a RMI qui nécessite pas un apprentissage.

3-Le Coût :

Pour CORBA, il n'existe pas des solutions gratuites et fiables comme les solutions propriétaires des fournisseurs, d'autre coté RMI est téléchargeable gratuitement et le Dotnet aussi mais qu'avec les outils de Microsoft (ex : VB Dotnet).

4-les langages de programmations :

RMI est programmable seulement avec JAVA. Dotnet les langages supportés par Microsoft. En revanche CORBA peut être programmé par n'importe quel langage.

5-plateformes et les systèmes d'exploitation:

Dotnet est pour les plateformes Windows. CORBA supporte tout type de plateformes

6-l'orienté objet :

RMI ne permet pas l'héritage multiple car il est pure java contrairement CORBA et Dotnet

6. Conception Client Serveur :

6.1/ Architecture centralisée :

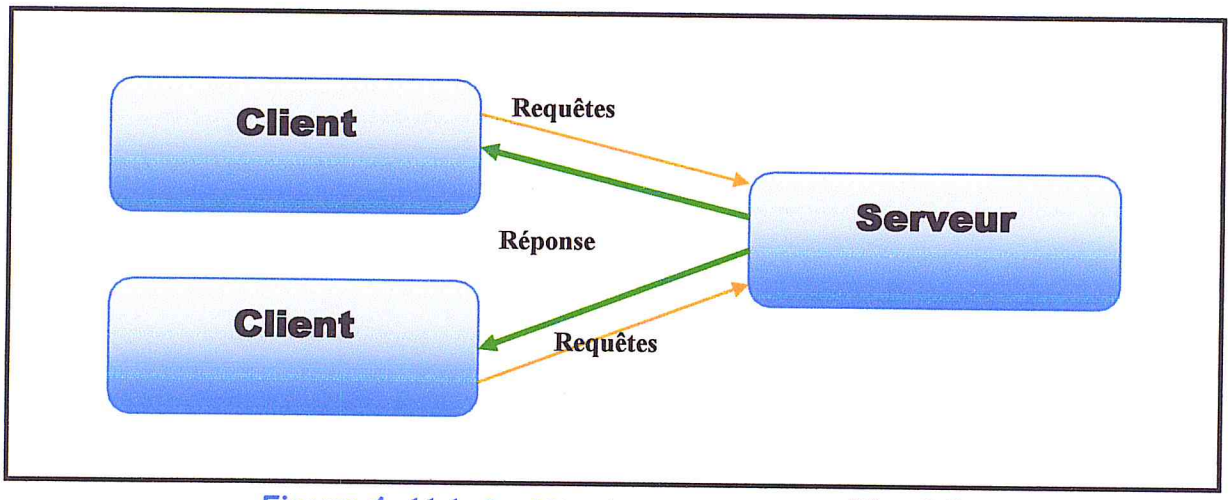


Figure 1. 11.1: **Architecture centralisée** [Geo01]

6.2. Architecture multi niveau :

6.2.1/Présentation de l'architecture a 2 niveaux:

Le serveur fournis des services demandés par le client directement, il ne fait pas l'appelle d'autre application extérieure.

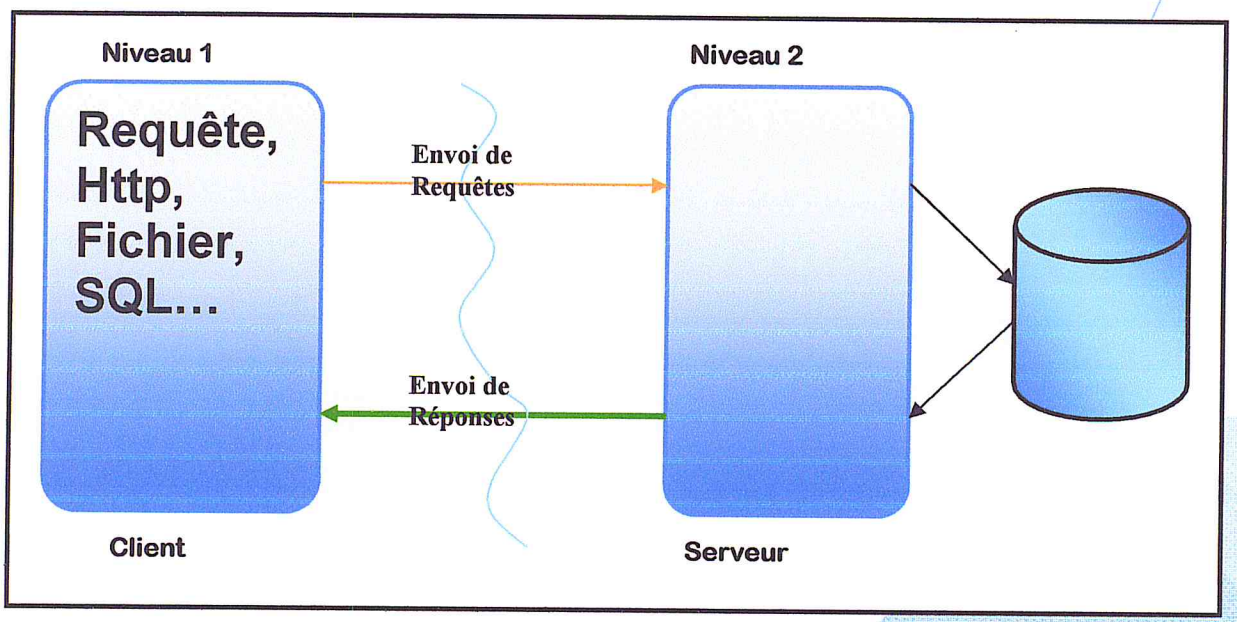


Figure 1.11.2 : **Architecture a deux niveaux** [Geo01]

6.2.2/ Présentation de l'architecture a 3 niveaux:

Le serveur faire appeler un autre serveur pour devenir capable de fournir des services au client en générale le deuxième serveur est serveur de base de données et le premier nommé ***un middelware***.

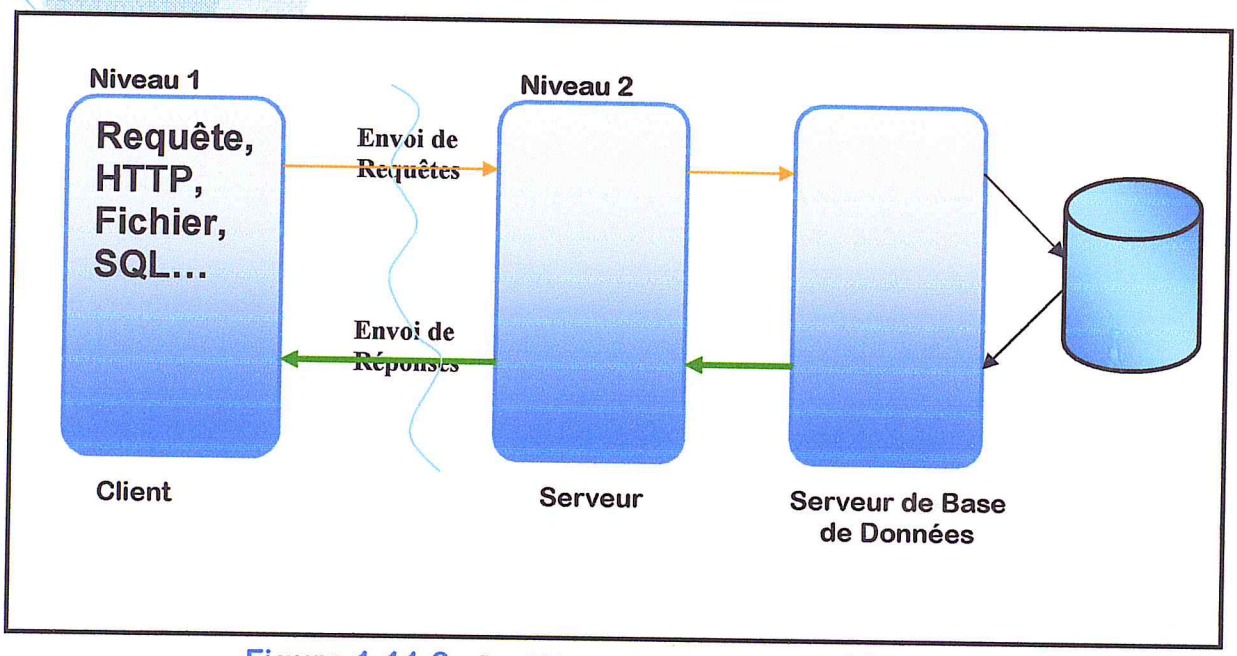


Figure 1.11.3 : **Architecture a 3 niveaux** [Geo01]

6.2.3/ Présentation de l'architecture a N niveaux:

Donc il y a des architectures a ***n niveaux*** selon les services demandés par le client et les spécialités des serveurs.

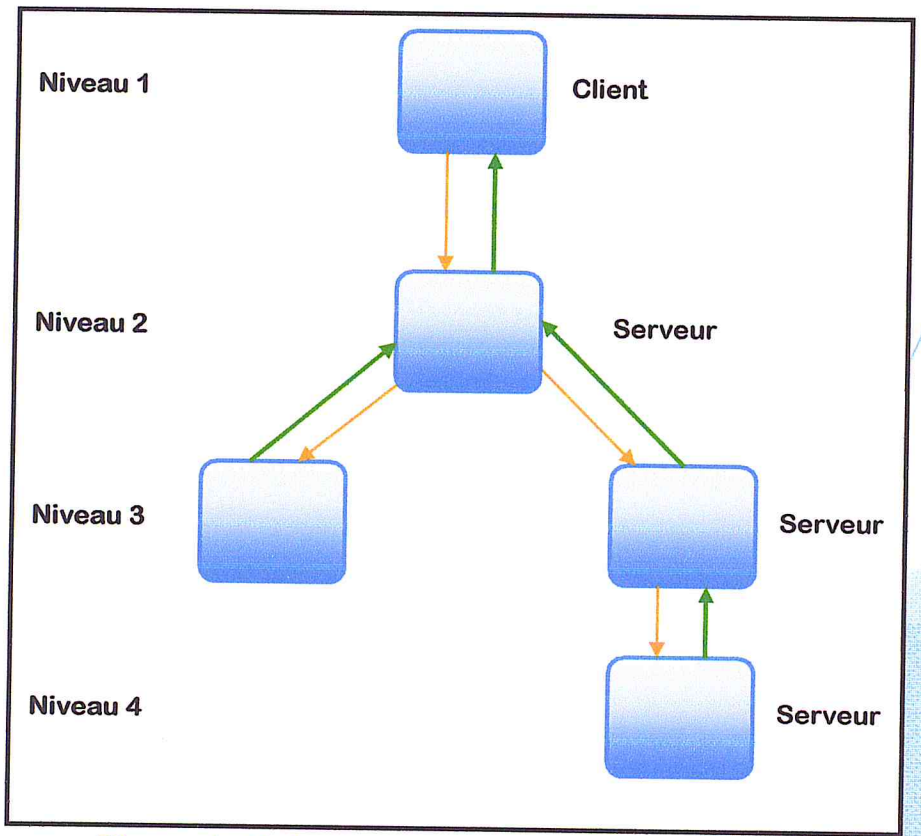


Figure 1. 11.4: **Architecture a N niveaux** [Geo01]

Peer To Peer

7. Le réseau pair à pair :

7.1/ Introduction :

Le réseau P2P ou ''Peer to Peer'' est l'équivalent d'un réseau ''poste à poste''ou ''pair à pair'' qui désigne une architecture de réseau où les postes (hôtes) sont connectés entre eux directement et partagent leurs ressources [PTP06], Tous les postes sont équivalents à la fois serveur et client.

7.2/ Définition :

Les P2P est une manière de manipuler l'information mais de ne pas servir des serveurs mais de relier tous les processus d'information en suppriment les serveurs qui sont des machine de grandes puissance alors coûteuse, alors l'utilisateur dans un pair a pair est a la fois client et serveur

7.3/ Topologie :

Le réseau pair à pair se greffe au réseau internet. Ils présentent donc une topologie virtuelle (overlay network).

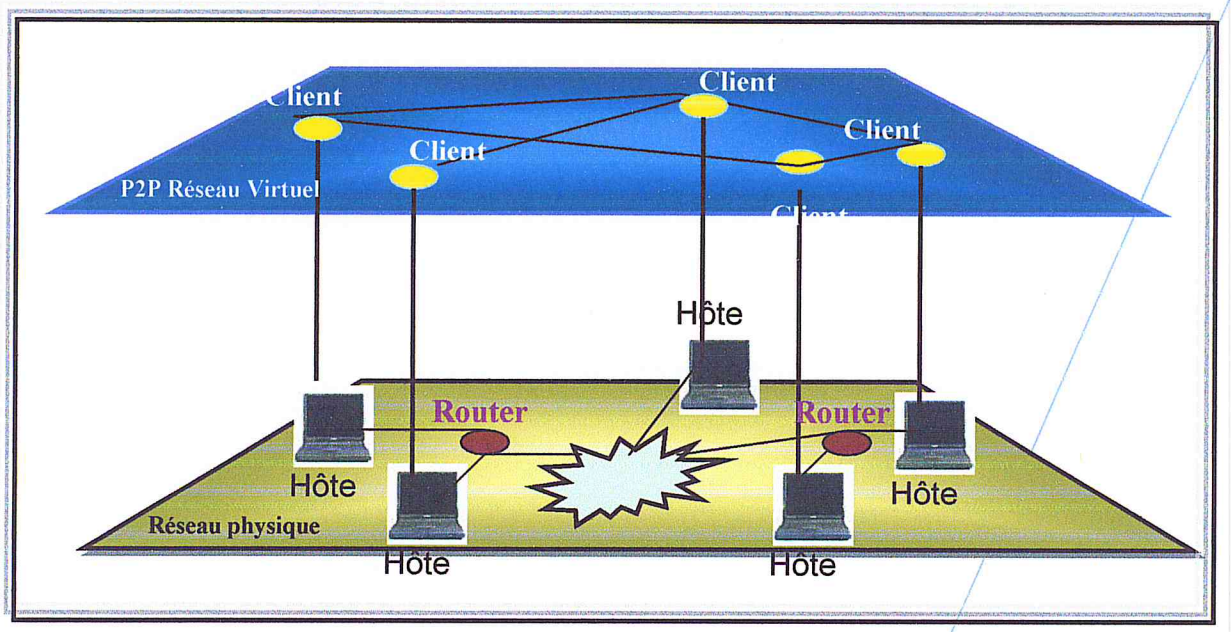


Figure 1.12 : Topologie des réseaux P2P [PTP06]

7.4/ Caractéristiques des systèmes P2P :

Les principales caractéristiques des différents protocoles sont:
- La localisation des fichiers dans un environnement distribué ;
- Des index du réseau P2P ;
- La libre circulation des fichiers entre systèmes ;
- Un mode de communication standard (TCP et http) ;
- Des capacités de connexion variables suivant les modèles ;
- Des peers non sûrs ;
- Aucun peer n'a une vue globale du système.

7.5/ Conception du modèle P2P :

7.5.1/ Le modèle P2P centralisé :

Modèle est entre le client/serveur et le pur P2P.
Le serveur est un intermédiaire de coordination ou/et de communication entre pairs et la consommation des ressources se fait par connexion directe.

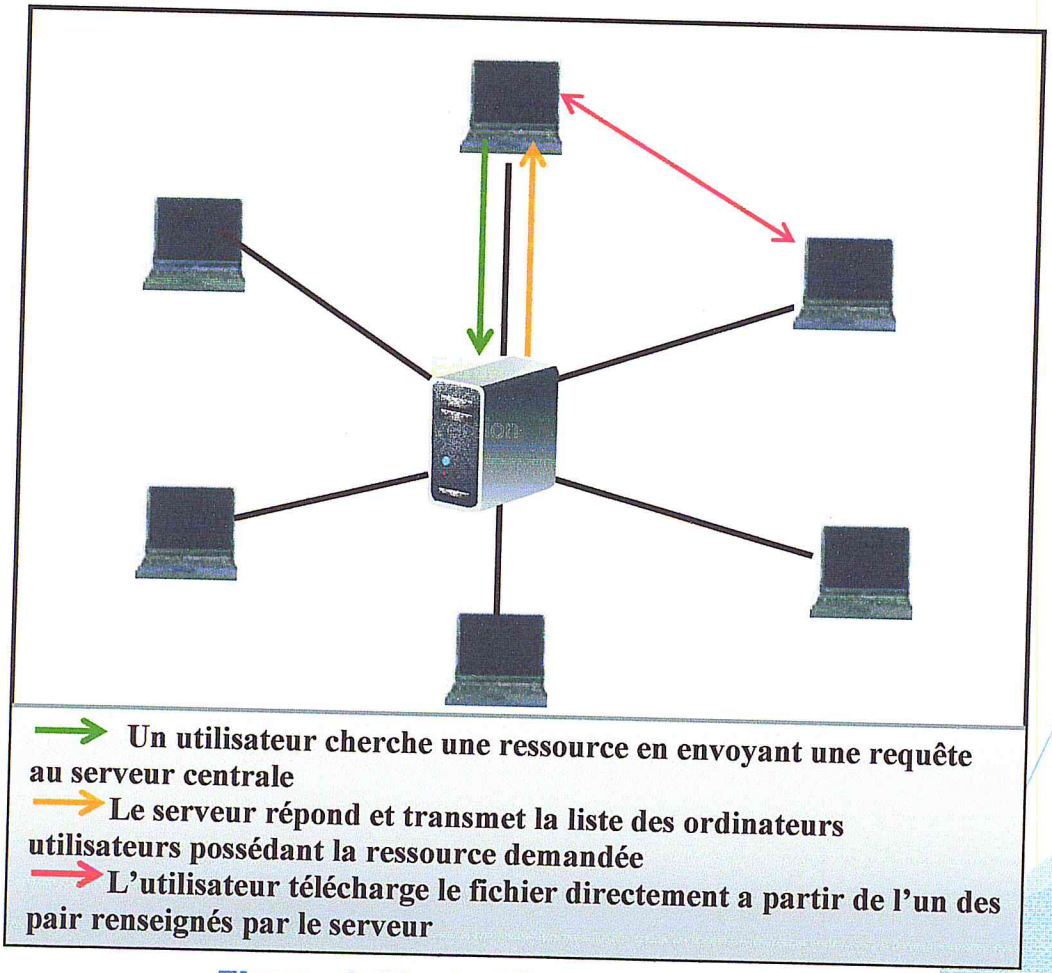


Figure 1.13 : Architecture Centralisée

7.5.1.1/ Les avantages :

1. Avantages habituels d'un serveur central
   - Facile à administrer
   - Facile à contrôler
2. Evite les recherches coûteuses sur le réseau
   - Efficacité de la recherche par indexation centralisée
   - Pas de routage
   - Planification de la gestion des utilisateurs
3. Tolérance aux fautes
   - Par un sondage régulier des peers connectés, état cohérent

7.5.1.2/ Les limites:

1. Pas d'anonymat : on est connus du serveur et des peers sur lesquels on télécharge
2. Limites habituelles d'un serveur central
   - Réseau complètement dépendant du serveur
   - Disponibilité
3. Problème du passage à l'échelle :
   - Saturation de la bande passante
   - Saturation du nombreux processus
   - Très facile de fermer le service car localisé
4. Mauvaises informations du débit des peers pour ne pas être sollicités

7.5.1.3/ Exemple (Architecture de réseau Napster) :

Partage de fichiers musicaux téléchargeables par les peer et assure la recherche effectuée sur l'index du serveur.

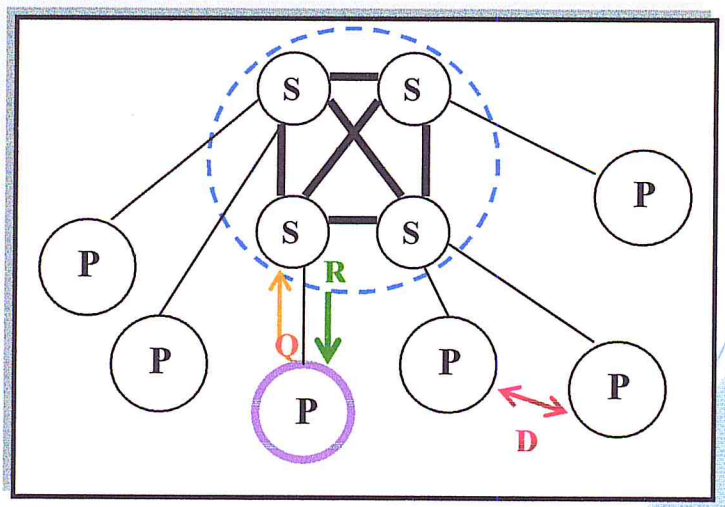


Figure 1.14 : Architecture de réseau Napster

7.5.2/ Le modèle P2P décentralisée :

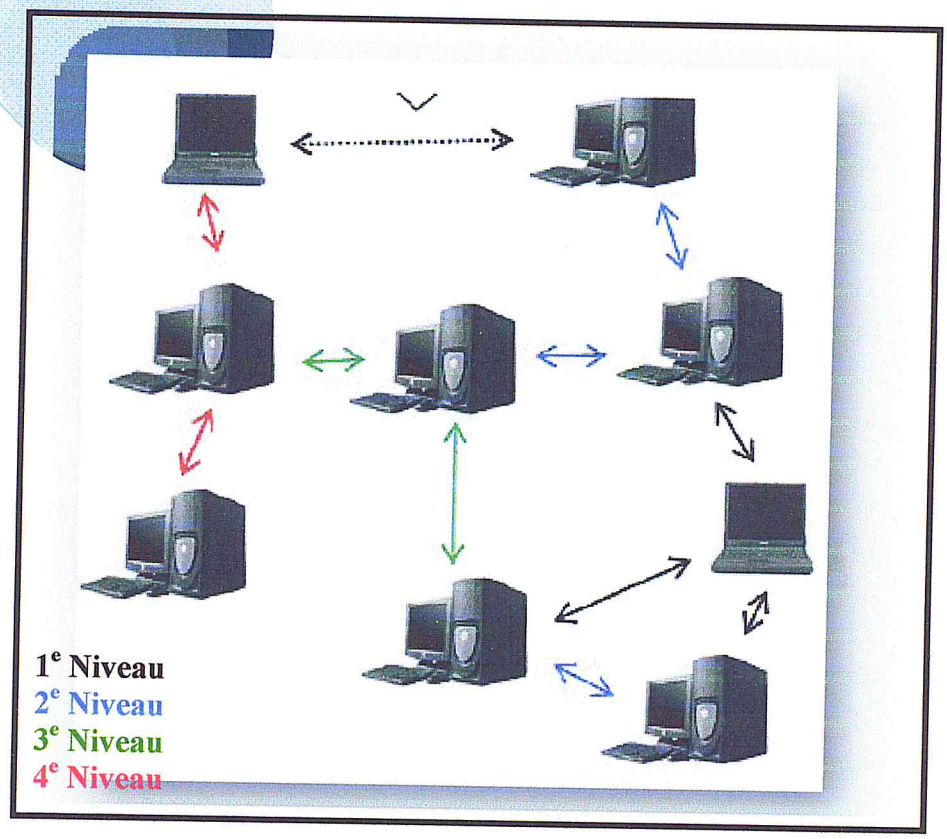


Figure 1.15 : Architecture décentralisée

Dans ce modèle, on observe l'absence du serveur intermédiaire car chaque peer possède un annuaire d'autres postes du réseau et les requêtes sont transmis et acheminés par les peers d'un poste à un autre respectant le paramètre de vie des requêtes ***TTL (Time To Live).***

7.5.2.1/ Les avantages :

- Administration simple et mutualisée
- Topologie évolutive (taille illimitée en théorie)
- Disponibilité du réseau.
- S'adapte bien à la dynamique du réseau (allées et venus des pairs)
- Total anonymat en théorie.

7.5.2.2/ Les limites:

- Gros consommateur de bande passante à cause des broadcasts envoyés.
- Pas de garantie de succès, ni d'estimation de la durée des requêtes (utilisation de TTL).

7.5.2.3-Architectures hybrides:

Ce modèle regroupe le modèle centralisé et le modèle pur peer to peer où les serveurs sont des supers peer voyant la figure précédente.

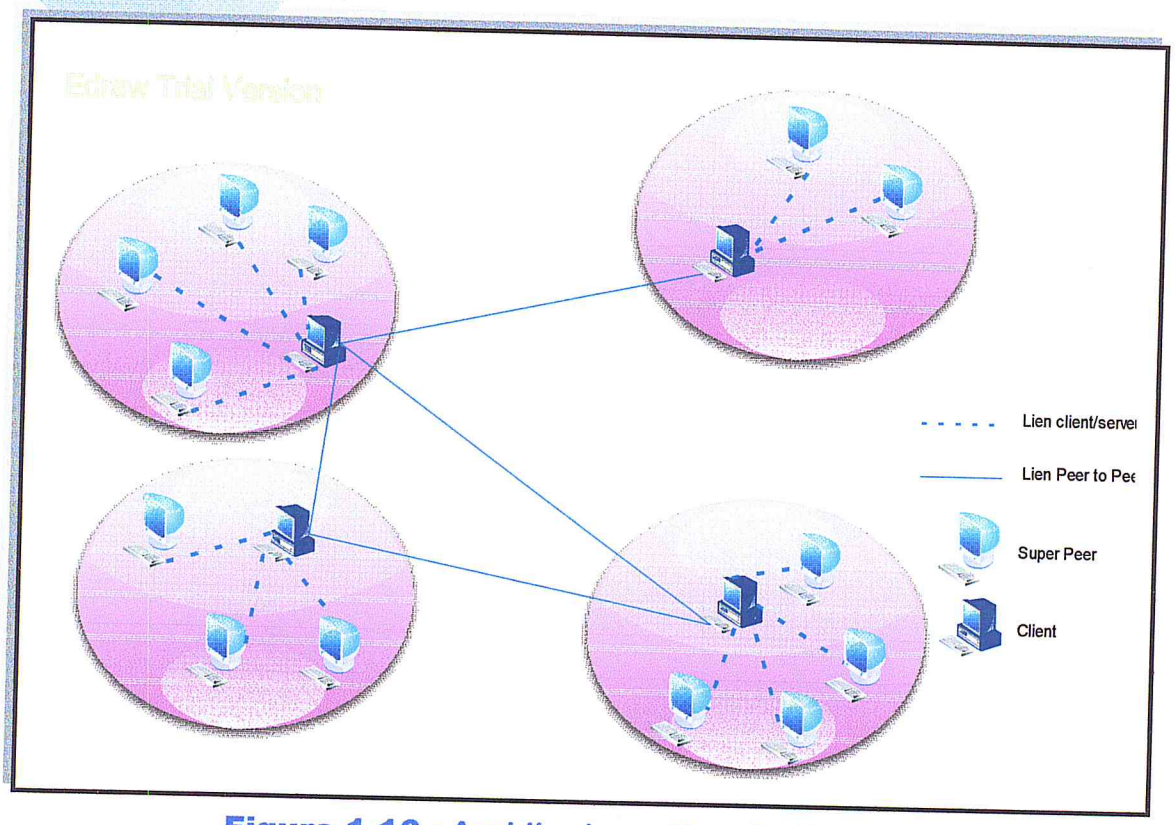


Figure 1.16 : Architecture décentralisée

7.5.2.4. Les Protocoles utilisés [PTP06]:

- **Peer Discovery Protocol**
  Pour découvrir les nœuds, les groupes de nœuds, les annonces ou toute autre ressource faisant l'objet d'un advertisement.

- **Peer Resolver Protocol**
  Pour l'envoi et la réception de requête et de réponses.

- **Peer Information Protocol**
  Utilisé pour recueillir des informations sur le statut ou les capacités de n'importe quel élément de la plate-forme (peer, peer group, pipes…);

- **Peer Membership Protocol**
  Régule les droits de création, d'entrée et de sortie des groupes de nœuds.

- **Pipe Binding Protocol**
  Utilisé pour définir la fonction d'un pipe pour les nœuds extérieurs;

- **Peer Endpoint Protocol**
  Permet à un nœud d'interroger un nœud routeur pour connaître une route.

7.6/ Comparaison entre Client Serveur et peer to peer :

Le P2P est une alternative au modèle client/serveur, qui est constitué d'un serveur ou d'un groupe de serveurs et de beaucoup de clients. Dans le modèle pur du P2P, il n'y a plus de serveur, tous les participants sont des peers.

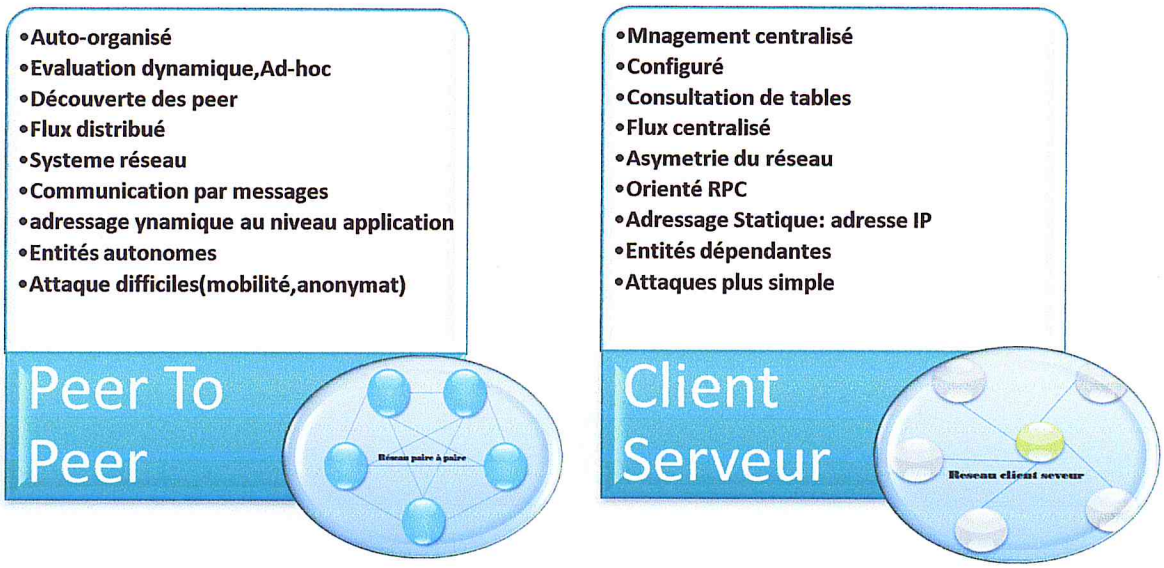


Tableau1.1: Comparaison entre pair à pair et client serveur

7.7. Avantages et limites :

7.7.1. Avantages du modèle Client/serveur:

- **Des ressources centralisées**: Le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs
- **Une meilleure sécurité**: Le nombre de points d'entrée permettant l'accès aux données est moins important.
- **Une administration au niveau serveur**: Les clients ayant ont moins besoin d'être administrés.
- **Un réseau évolutif**: Grâce à cette architecture on peut supprimer ou rajouter des clients sans perturber le fonctionnement du réseau avec peut de modifications.

7.7.2/ Inconvénients du modèle Client/serveur:

- **Un coût élevé:** vu le coté du serveur.
- **Le nombre limité** des Client Connecter au serveur simultanément
- **Un maillon faible**: Le serveur est le seul maillon faible du réseau client/serveur, C'est le cœur du réseau.
- **Seul le client présente une application** mais le serveur répond aux demandes des clients, c'est-à-dire que les réponses rendues sont construites indépendamment du client alors une partie de ces réponses est inutile se qui augmente le trafic sur le réseau de plus se modèle a besoin d'une connexion permanente sur le réseau

Synthèse:

1. Introduction :

Le modèle client serveur est utilisé pour construire la quasi-totalité des applications distribuées cependant, ce modèle arrive à une étape ou il peut plus parcourir les exigences du développement et des nouveaux besoins tel que : Internet car avec l'émergence de certaine application complexe sur le net le modèle client serveur ne semble pas être la solution idéal pour combler les exigences d'utilisateur en terme de qualité de réponse en temps et contenu. Ainsi que la difficulté de connexion entre hôte sur un système mobile ou un réseau sans fils car le réseau client serveur risque d'être saturer la majorité du temps et aussi cour un problème de sécurité qui n'est pas complètement assurer sur se type de système.

2. Proposition:

Avec l'émergence des d'applications réparties : Le modèle client serveur devra être adapté aux nouvelles exigences de cela une extension du modèle de communication client serveur peux être réalisé a travers la notion de mobilité du code comme suit :

2.1/Mobilité par requête :

Le client et le serveur ne se déplace pas réellement car il lance des requête que serveur :

1 /le client lance une interrogation au serveur.

2 /le serveur retourne une synthèse de ces résultat.

2.2/ Mobilité réelle :

Le client se déplace réellement à travers le code sur différents sites serveurs qui doivent le reconnaître à travers les plateformes unifiés.

Conclusion:

Sur la partie qui suit nous allons mettre au point une étude détailler sur les différent types de mobilité que nous allons juger par la suite de leurs apport par rapport au modèle client serveur.

Chapitre 2:

Les systèmes distribués : Technologie Code Mobile

1. Introduction :

Vue le modèle client serveur aborder précédemment on a pu conclure que le serveur offre seulement des services selon la demande du client qui lui seul représente une application, alors ces réponses sont indépendante et ainsi le modèle peux causer un flux inutile sur le réseau a cause des réponses qui englobe qui ne sont pas sélectionnée.la perte de connexion est un facteur qui semble pausé des problèmes majeurs qui influes sur la fiabilité du modèle clients serveurs

Afin d'aboutir a des répondes et des solutions a ses contraintes, la mobilité va être entamé dans la partie qui suit :

La mobilité c'est la migration d'un programme tout en gardent son état initiale, elle sert à rendre le programme plus fiable et aussi le system distribuer pourra mieux fonctionner. Cette opération peut se faire de la manière suivante:

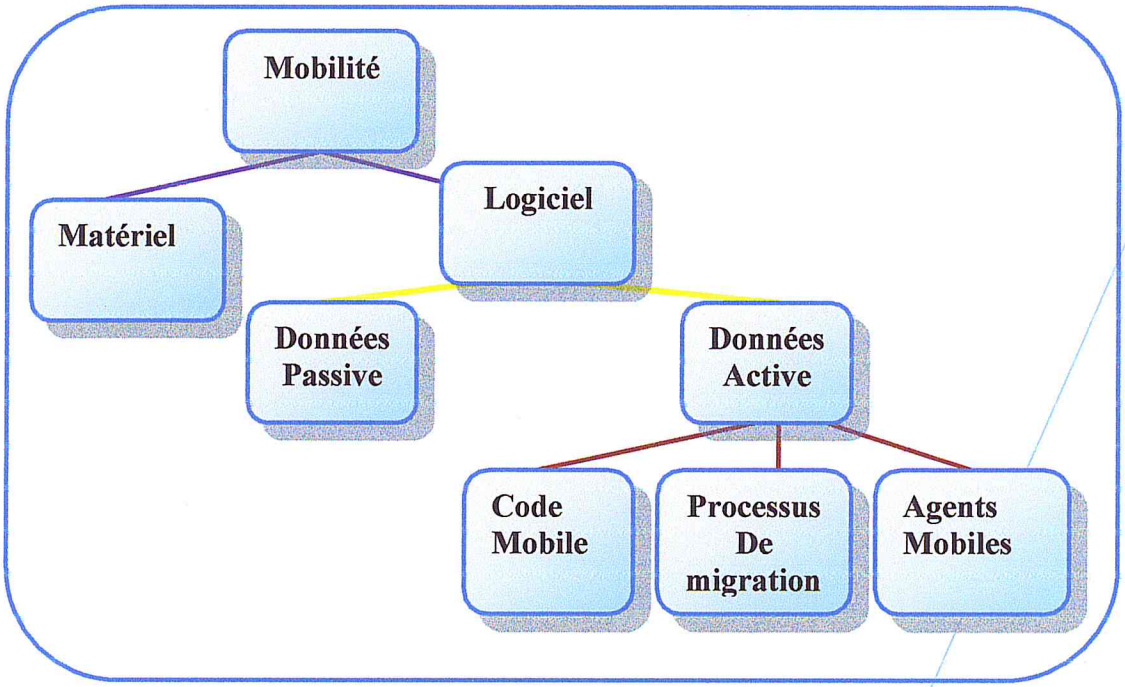


Figure 2.1 : Arborescence de la mobilité [Mil07]

Généralement la mobilité peut être composée en deux sortes :

Mobilité matérielle (hardware) : Traite la mobilité du matériel informatiques mobiles comme la restriction sur la sur la connexion d'ordinateurs portables, et des IP mobile

Mobilité logicielle (software) : la sécurité, la localisation, La dénomination, la transmission et de la communication. Cette dernière représenté par :
Données passives représente les méthodes traditionnel transfère de données entre deux machines
Données actives qui peuvent être classifié en trois parties :

1/Code mobile : transfère juste le code entre les machines tel que java applet
2/ migration de processus: adapte le transfère du code et des données, et aussi il adapte transfère de l'autorité tel que la l'accessibilité a un fichier système télécharger, mais cette accessibilité est sous le contrôle d'un administrateur.
3/Agents mobiles : transfère le code les données et l'autorité qui agir pour les droites des propriétaires dans un large accès. Exemple l'utilisation de l'internet.

Ces trois parties représentent évolution incrémentale de l'état de transfère et dont on va détailler plus bas.

2. Le code mobile:

C'est un ensemble de programme (code seul) capable de ce déplacer pendent leur exécution, il est exécuté de manière partielle ou parallèle sur une très grande diversité d'hôte. Ce déplacement peut ce faire de deux façons:

2.1. Code à la demande:

C'est un terme général pour toute la technologie qui envoie un logiciel exécutable à partir d'un serveur à un ordinateur client à la demande de la part du client du logiciel (par exemple, le navigateur) et pour le code mobile le code est demander (d'un site distant) pour faire son exécution sur la machine locale.

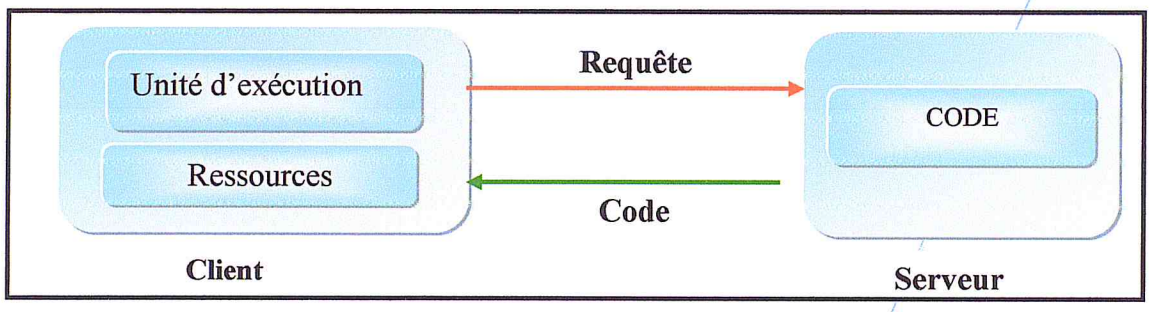


Figure 2.2 : Code a la demande

Code à la demande est une utilisation spécifique de code mobile. Un exemple bien connu pour le code à la demande de paradigme sont les applets Java: une applet du code du programme est inactive sur certains serveur Web jusqu'à ce qu'un utilisateur

(client) demande une page Web qui contient un lien vers l'applet à l'aide du client de navigateur Web. Lors de cette demande, la page web et l'applet sont transportés à l'utilisateur de la machine à l'aide de HTTP. Lorsque la page est affichée, l'applet est lancé dans le navigateur et exécute localement, à l'intérieur de l'ordinateur de l'utilisateur jusqu'à ce qu'il soit arrêté (par exemple, en laissant à l'utilisateur de l'applet de la page Web). Cela termine le cycle de vie de l'applet.

2.2. L'évaluation à distance : l'évaluation à distance est un terme général pour toute la technologie qui implique la transmission de logiciels exécutables à partir d'un ordinateur client à un serveur informatique pour exécution ultérieure sur le serveur. En effet on transfère un code et ces donné initialisation vers un site distant qui va utiliser ces ressources pour exécuter le code, Lorsque le programme a pris fin, les résultats de son exécution sont envoyés au site locale.

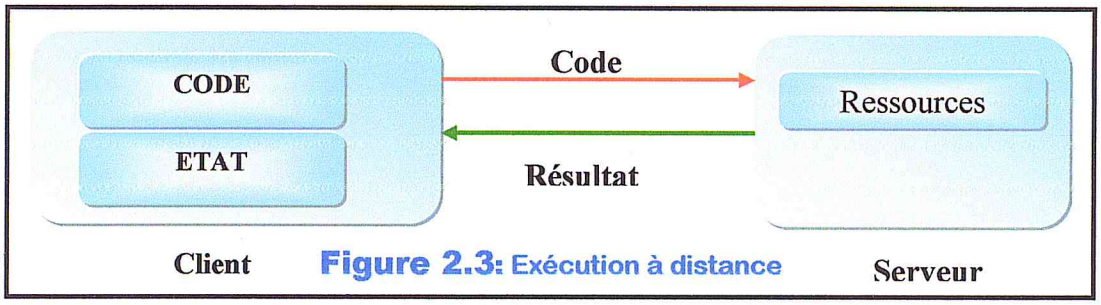


Figure 2.3: Exécution à distance

3/ migration de processus:

3.1/ C'est quoi un processus [DOC01]: Ensemble d'activités interactives qui transforme des éléments d'entrée en éléments de sortie. C'est aussi un système d'exploitation qui représente l'abstraction d'une instance d'un programme d'ordinateur en cours d'exécution.

Le processus de migration est l'acte de transfère de processus entre deux machines toute en permettent la répartition des charges, et vue sa résilience il facilite l'administration des systèmes et l'accessibilité des informations localisé. La figure 4 représente une architecture d'un système de migration

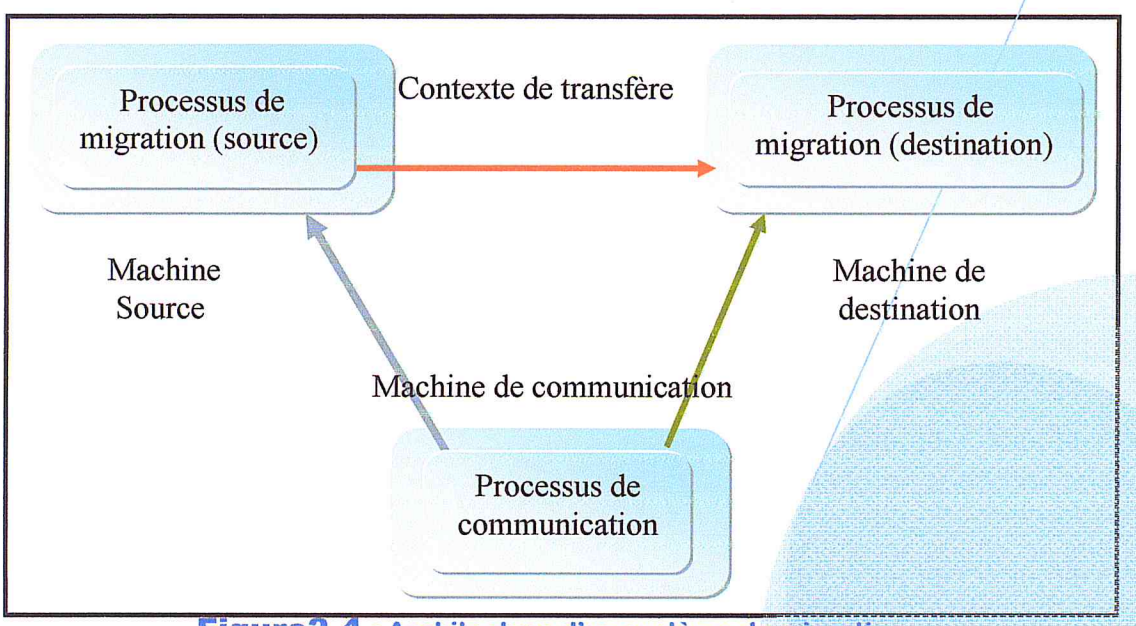


Figure2.4 : Architecture d'un système de migration

Processus de migration consiste à extraire l'état du processus dans la machine source et le transférer a la machine destinataire ou une nouvelle instance est crée, la mise a jour des connexions avec d'autre processus sur la machine de communication, il existe deux types de migrations :

- La migration faible:

Il permet a un agent de ce déplacer quelque soit son état d'exécution et de reprendre cet état la ou il en été avant sa migration

- La migration forte:

La migration forte, quant à elle, permet de transfert du processus et ses données et l'état du processus, mais cette migration peut causer des pertes des traitements en cour.

3.2/Objectifs [MIL07]:

Les objectifs du processus de migration sont étroitement liés au type d'applications qui utilisent la migration. Les objectifs du processus de migration incluent:

3.2.1/ Accès à plus de puissance de traitement :

Est un objectif de la migration quand il est utilisé pour la répartition de charge. La migration est particulièrement important dans les algorithmes *récepteur de l'initiative* distribuées d'ordonnancement, quand une machine un peu chargé annonce sa disponibilité et son processus de migration initiés surchargé d'une machine distante. Cela permet aussi de partitionner dynamiquement la charge dans un réseau en migrant toujours vers les machines les plus puissantes et les moins chargées. Et cela lui permet de profiter des ressources de la machine distante comme l'espace disque, la mémoire utilisée, et même la puissance de calcul du processus.

3.2.2/ L'exploitation des ressources localisés :

est un des objectifs de la migration, Dans les cas où il est plus efficace d'accéder à des ressources de distance d'un processus à l'autre bout de a communication, Le déplacement du processus à la machine sur laquelle s'exécute l'application avec laquelle il communique, ou sur la machine sur laquelle se trouve l'information, améliore sensiblement la performance d'un réseau vue qu'il communique avec des applications ou récupère des données localement en rendant l'exécution du processus plus rapide .

3.2.3/ Disponibilité du système:

On déplace les applications des machines susceptibles de tomber en panne sur d'autres machines pour garantir la bonne marche de l'exécution de ces applications.

3.2.4/ Administration du système:

Grâce à la migration des processus, on doit assurer l'exécution des applications sur une machine même si des travaux d'administration nécessitent la modification des paramètres d'une machine.

3.2.5/ Calcul mobile:

On peut effectuer aussi bien le transfère entre deux machines que que l'une possède une puissance de calcul plus grande que l'autre mais qui ne dispose pas de connexion réseau entre ces deux machines.

3.3/ Algorithme de Migration : [MIL07]

Bien qu'il existe de nombreuses migrations d'implémentation et les dessins, la plupart d'entre eux peuvent être résumées dans les étapes suivantes :

1. **A la demande de migration est délivré à un nœud à distance.**

Après négociation, la migration a été acceptée.

2. **Un processus est détaché de sa source** par le **sus-nœud** : dans l'attente de son exécution, déclarant être en migration état, et de réorienter temporairement la communication en tant que décrit dans l'étape suivante.

3. **Communication est temporairement redirigé** accueillement jusqu'à l'arrivée des messages adressés, et par les acheminer vers le processus après le migration. Cette étape se poursuit en parallèle avec les étapes 4, 5 et 6, tant il existe d'autres les Messages. Une fois les canaux de communication sont activés après la migration (à la suite de l'étape 7), le processus de migration est connu pour le monde extérieur

4. **Le contexte d'exécution du processus est extrait,** le contenu de ce contexte étant dépendant de la stratégie de migration choisie.

5. Un **nouveau processus est créé sur le site destination** ; le contexte d'exécution extrait sera ultérieurement affecté à ce processus et ce processus représentera le processus déplacé.

6. **La communication est redirigée** en renvoyant les messages au nouveau processus pour Que ceux-ci soient traités après la migration.

7. **Le contexte d'exécution extrait** est envoyé au nouveau processus pour y être intégré .Pour des raisons de performance, certains mécanismes de migration n'exigent pas le Transfert de tout le contexte d'exécution avant la fin de la migration.

8. **Le nouveau processus continue l'exécution** dès que la partie de son contexte nécessaire à son exécution est reçue. À partir de ce moment, la migration prend fin. Une partie du contexte du processus sur le site source peut être maintenue, dans le cas d'un transfert retardé ; une fois tout le contexte transmis, le processus sur le site source peut être définitivement arrêté.

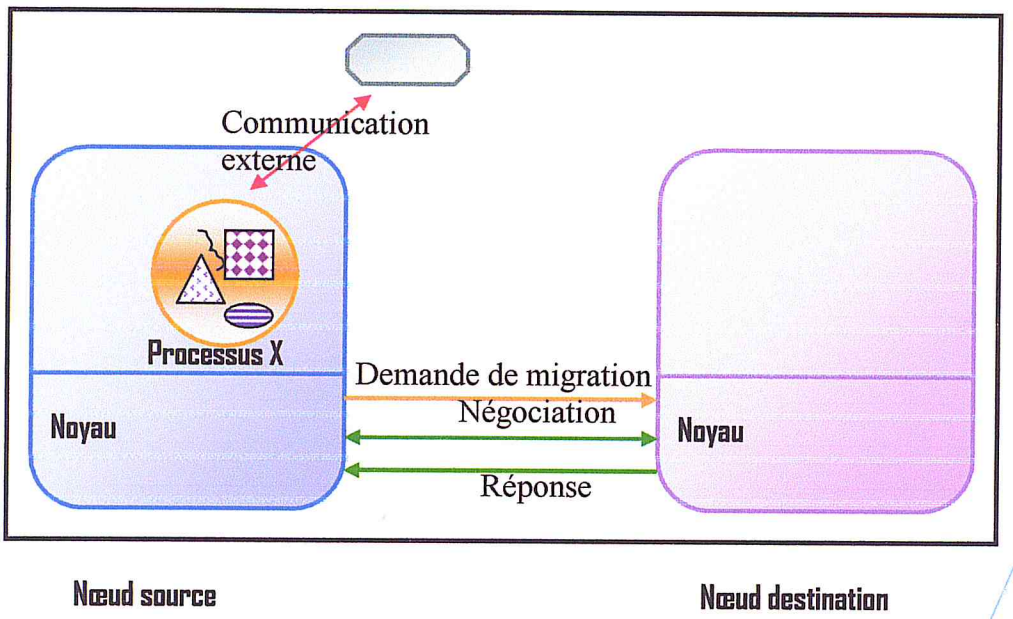


Figure2.5.1 : Envoi d'une requête de migration vers une machine distante

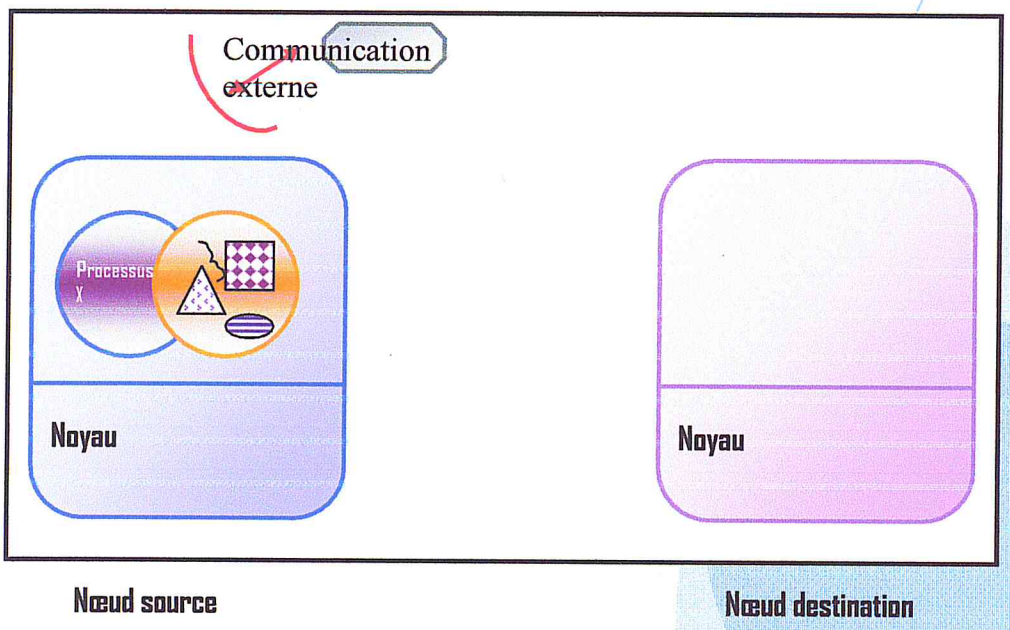


Figure2.5.2 : Le processus se détache de son nœud source

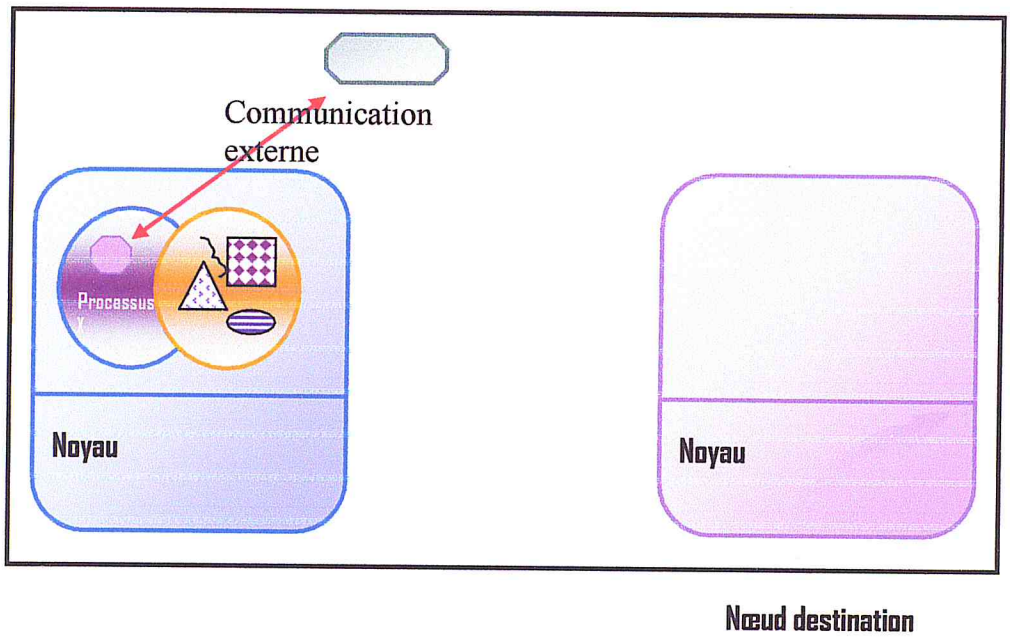


Figure2.5.3 : Les communications sont temporaire redirigés (Fin de l'opération a l'étape 7)

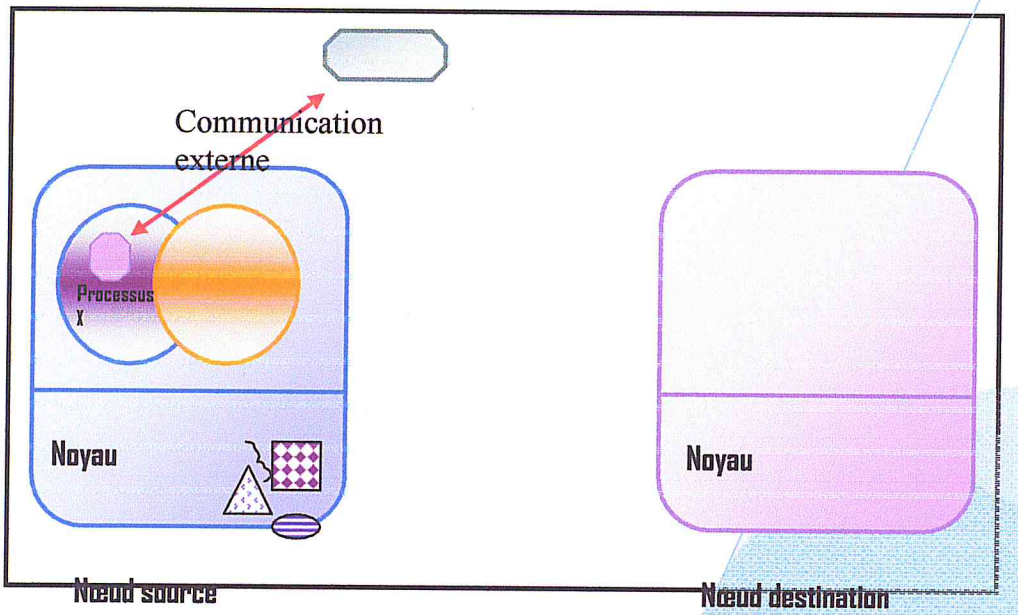


Figure2.5.4 : Extraction de l'état du processus

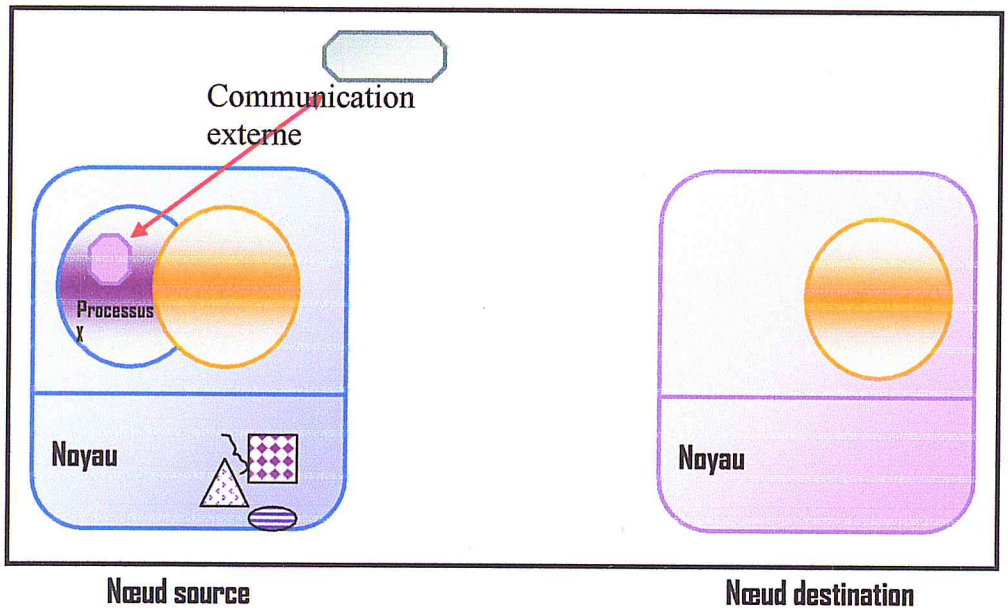


Figure2.5.5 : La création du processus distant

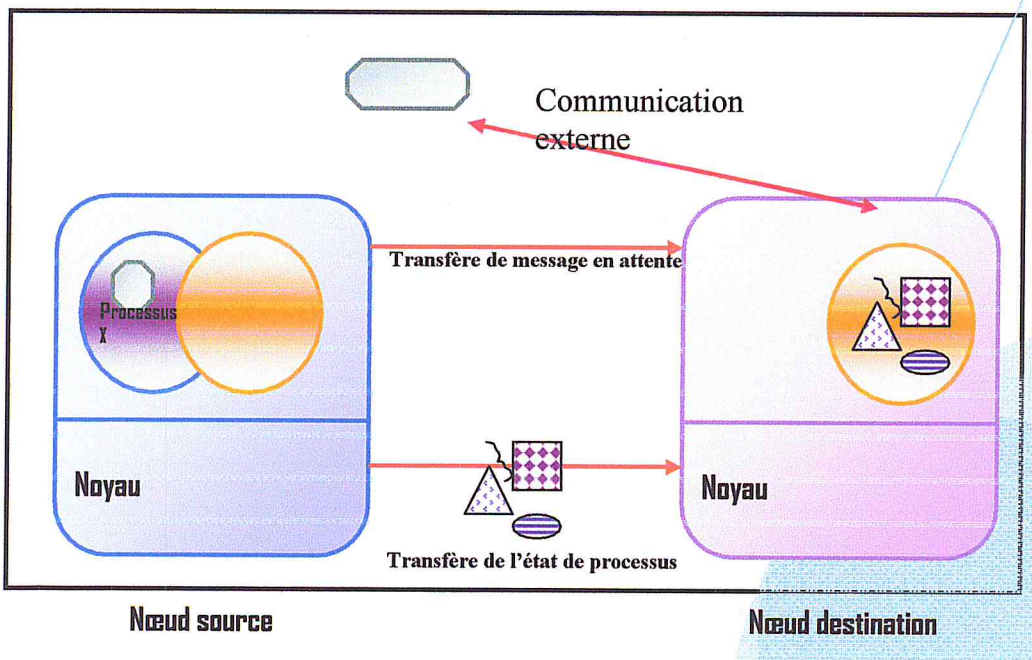


Figure2.5.6 : Transfère de l'état du processus

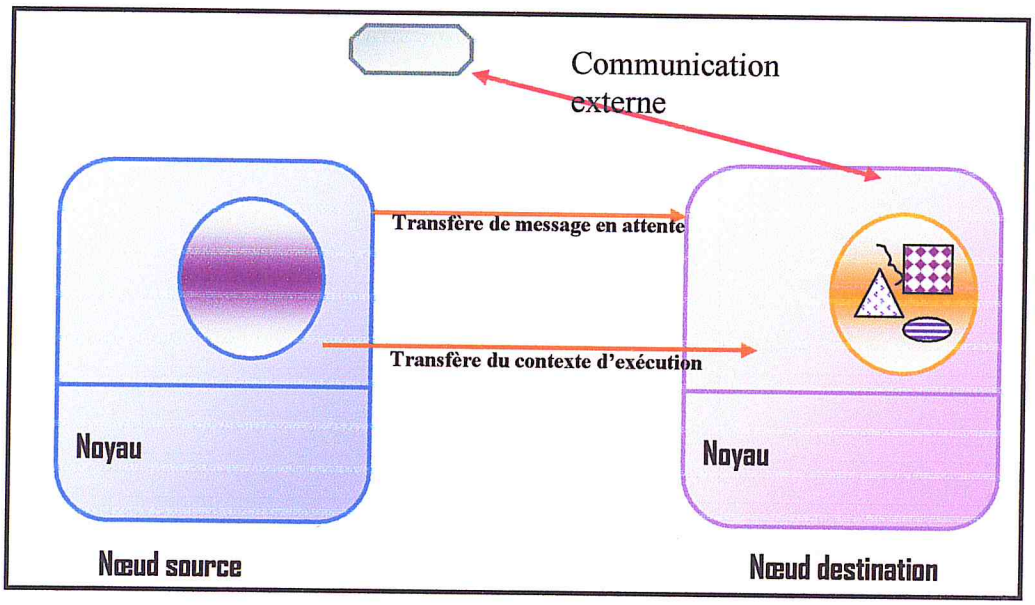


Figure2.5.7 : La création du contexte d'exécution

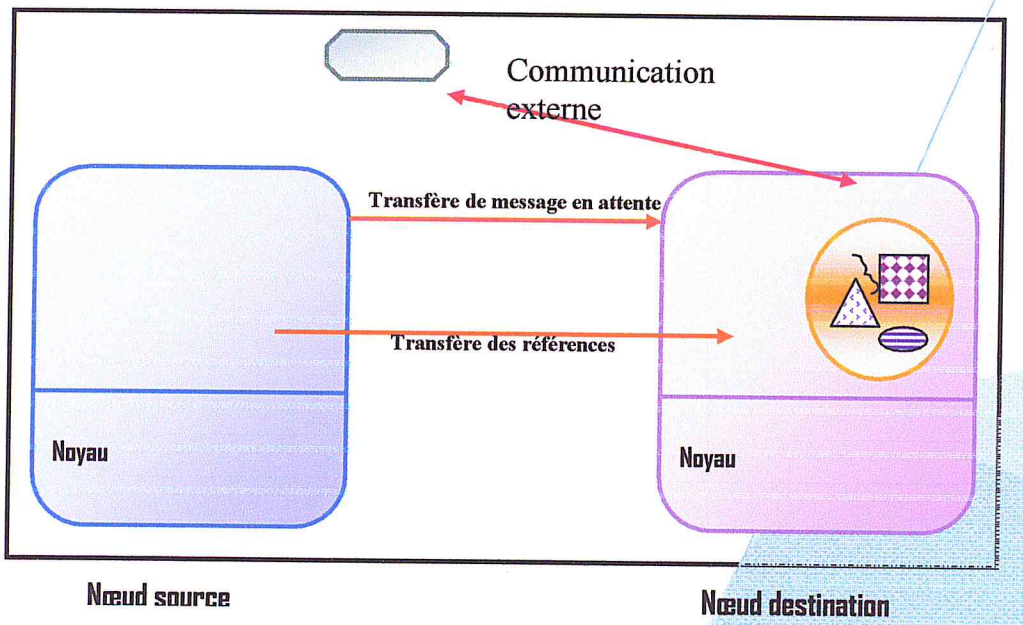


Figure2.5.8 : La fin de migration

3.4/.Problème de migration : [MIL07]

Vue quand parles souvent des systèmes distribuer alors on est confronté à plusieurs problèmes par exemple l'hétérogénéité des systèmes.

- Hétérogénéité:

Problème: Le problème de l'hétérogénéité c'est incompatibilité que se soit au niveau machine ou code car si la représentation d'une machine ne peut pas être comprise par une machine d'un type différent. Ce problème peut être résolus par : un format de traduction qui peut être compris par toutes les machines alors en doit mettre en place un traducteur a chaque extrémité de transfère .une deuxième solution qui consiste a mettre un traducteur au format de la machine distante alors on aura autant de traducteurs les communications, dans ce cas cette présentation est moins couteuse que la première.

- Qualité d'interaction entre processus [MIL98]:

Problème: lorsqu'un message arrive à destination d'un processus qui est en cycle de migration, le message peut être perdu sans modification.

Qui peut être traité par la création d'un nouveau état au processus qui prévoie d'être en migration autrement dit il ne peut pas recevoir des messages. Donc les processus doit signaler son nouvel état avant de commencer la migration cette information est transmise a tous les autres processus avec qui il a une communication. Ou bien le ou les messages sont retenus sur un file d'attente jusqu'à déclaration que le processus est sur sa nouvelle destination et la les messages peuvent être envoyés.

- Qualité d'interaction sur les canaux:

Problème : Lors d'un déplacement du processus d'une machine a une autre, a travers un canal qui va avoir une descripteur ouvert pour la machine source et qui n'aura aucun sens pour la machine sur la machine destination, la résolution de ce conflit est très importante lorsque la machine source tombe en pane ou elle doit être réinitialisée et pour cela en crée un lien de poursuite permettent au processus migrants d'avoir un accès aux canaux ouvert même après la migration et aussi en peut adapter une représentation commune des canaux ouverts, avec cette solution un canal ouvert possède toujours le même descripteur dans le système réparti.

4. Les agents Mobiles :

4.1. Qu'est ce qu'un agent mobile ? [MEM02]:

- **Définition 1 :** un agent est une entité logiciel ou physique a qui attribuée une certaine mission qu'elle est capable d'accomplir de manière autonome et en coopération avec d'autres agents [2]

- **Définition 2 :** Les agents mobiles sont proposés comme un mécanisme pour la recherche d'information ou le commerce électronique.

- **Définition 3:** Un agent mobile est un code mobile capable de se déplacer de manière autonome d'une machine a une autre ou d'un site a un autre il transfère son code, ses données et son état d'exécution, pour accéder à des données ou à des ressources ou bien pour faire exécuter des tâches, il est réactif, proactif peut s'adapter a son environnement [MEM02].

4.2. Architecture d'un agent mobile [MEM02] :

Les agents mobiles sont des entités composées des différentes parties, la première partie est le code qui constitue les instructions définissant le comportement de l'agent la seconde partie est l'état d'exécution de l'agent. Et la troisième c'est Les ressources et les données utilisées par l'agent mobile.

Malgré le fait que ces deux parties (le code l'état d'exécution) soient indépendantes, au moment de la migration l'agent les captures les empaquettent pour les prendre, car si l'un lui définit son comportement l'autre lui permet de se rappeler ses tâches précédentes.

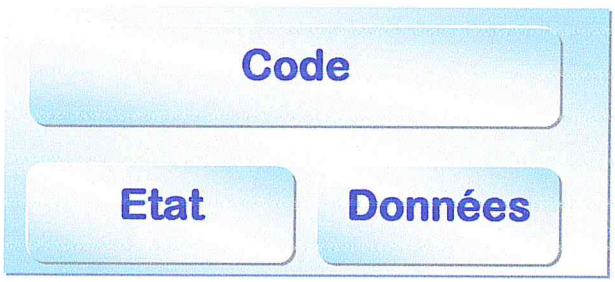


Figure 3.1: Structure d'un agent mobile [Bel04]

Exemple :

Dans le cas d'utilisation du langage de programmation JAVA les classes définissant l'agent sont les codes, et les données nécessaires à l'exécution de l'agent sont une partie de l'état de l'agent.

4.3. Classification des grands types d'agents mobiles [CRO05] :

Nous allons opter pour une classification des agents mobiles selon certain critères qui sont empruntés à la classification des processus système qui peut donner une référence généralisée pour les concepteurs mais qui n'est pas figé. Cela nous aboutis à deux types d'agents mobiles :

Légers :

Qui on pour caractéristique de se déplacer plus souvent et plus rapidement a cause de leurs mission qui na pas pour but de faire de longue traitement sur les sites

Lourds :

Se déplace souvent et lentement a cause de leurs mission qui a pour but de faire de long traitement sur les sites.

4.4. Caractéristiques d'un agent mobile [DAN98]:

4.4.1. La mobilité :

Est la migration d'un programme avec restauration de son état initial, elle sert à augmenter la fiabilité et le bon fonctionnement des systèmes distribués.

4.4.2. Réduction de la charge dans les réseaux [DAN98].

les agent mobiles sont envoyés sur le site ou il ya une exécution de taches alors les messages échangés deviennent locaux et de telle une libération de charge sur le réseau

L'utilisation des agents mobiles dans les réseaux permet d'affranchir le temps de réponse et la réduction des délais de communication

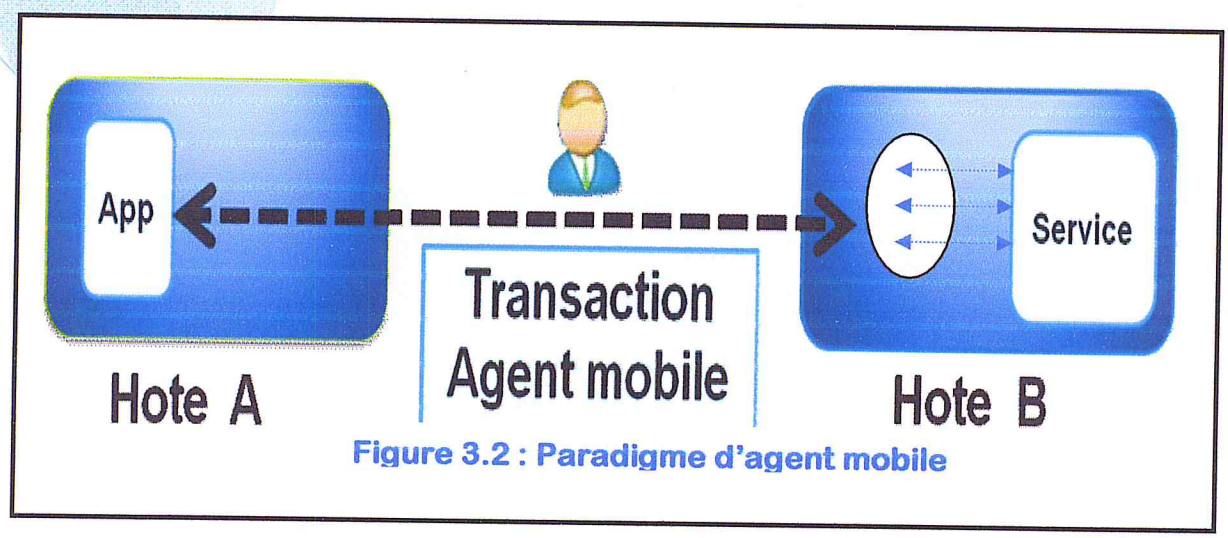


Figure 3.2 : Paradigme d'agent mobile

4.5. Exécution asynchrone et autonome :

Dans le paradigme agent mobile, un agent peut être envoyé sur le terminal mobile au moment de la connexion et travaille sur ce dernier après avoir interrompu la connexion, le même agent peut attendre la prochaine connexion pour envoyer des informations sur le réseau signifiant par exemple qu'une tâches a été terminer.

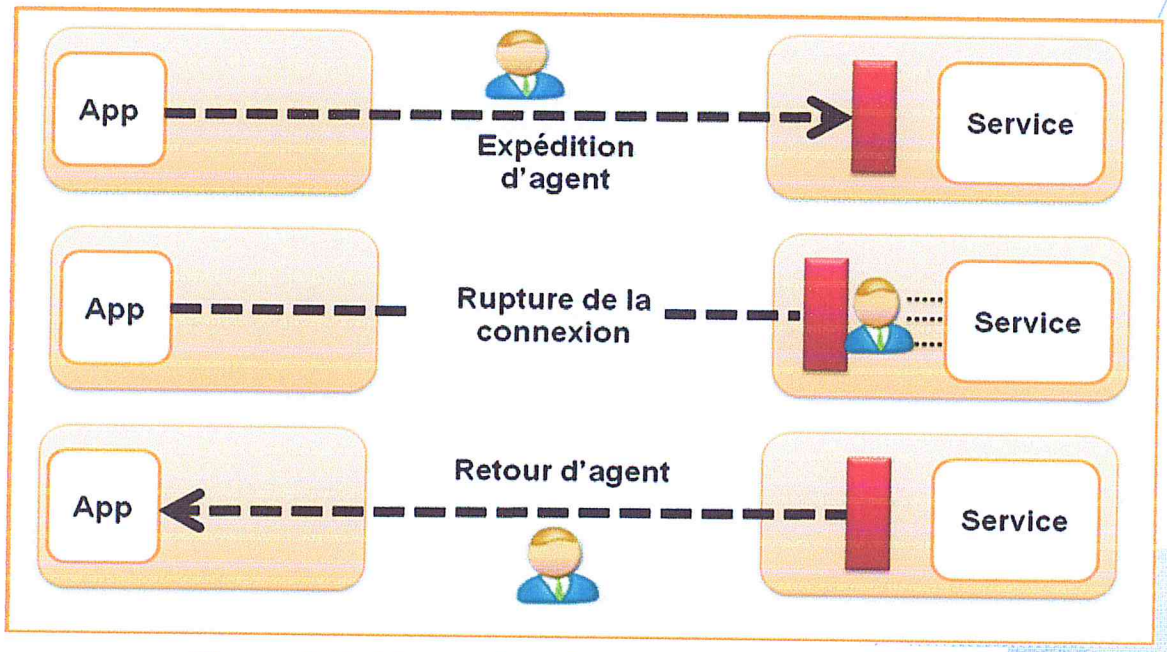


Figure 3.3 : L'asynchronisme des agents mobiles

4.6. Adaptation dynamique à l'environnement :

Les agents mobiles savent détecter les changements dans leur environnement, y réagir de manière autonome et s'adapter en conséquence. Ils ont la propriété de se répartir uniformément entre les machines d'un réseau pour résoudre un problème complexe de manière optimale.

4.7. Robustesse et tolérance aux fautes :

La propriété des agents mobiles à réagir de manière dynamique aux situations et évènements peu favorables, permet la conception des systèmes répartis robustes et tolérants aux fautes, lorsqu'une machine hôte est en difficulté, les agents mobiles visiteurs prévenus ont la possibilité de se dispatcher ailleurs dans le réseau, par contre dans le modèle client/serveur si le serveur tombe en panne, la session de travail collaboratif ne peut plus continuer.

4.8. Communication [EFB04]:

Dans le modèle client/serveur lorsqu'un message passe d'un client à un autre, deux transactions sont ainsi nécessaires, du premier client au serveur, puis du serveur au second client, mais les systèmes basés sur les agents peuvent interagir avec d'autres agents pour atteindre leurs buts locaux et globaux, donc la communication permet la coopération et la coordination entre eux.

4.9. La migration d'un agent mobile [DTH02] :

Un agent part d'un site initiateur à travers un support de communication comme un message puis visite un ensemble d'hôtes pour y effectuer un traitement donné. Finalement l'agent revient sur le site de départ pour rapporter ces résultats, il existe deux types de migration :

La migration forte:

Offre la possibilité du déplacement de l'agent quelque soit l'état d'exécution dans lequel il se trouve et de reprendre l'exécution après la migration exactement la ou elle en été avent.

La migration faible :

C'est le transfère de l'agent avec son code et ces données mais avec perte des traitements en cours.

5/ Cycle de vie d'un agent mobile [MEM02] :

Un agent possède plusieurs états par lesquels il peut passer, ces états constituent son cycle de vie (figure 3.4)

5.1. Création : un agent est crée soit par sa plateforme, soit par clonage.

5.2. Clonage : un agent peut être cloné de deux manières différentes

- **Clonage locale** : dans le même environnement (plateformes).
- **Clonage distant** : un agent dans un environnement A se clone dans un autre environnement.

5.3. Dispose : un agent peut être détruit soit par programmation(Automatiquement) ou par l'environnement

5.4. Stocké : l'agent est arrêté, mais pas détruit.

5.5. Migration : l'agent peut migrer d'un environnement A vers un autre environnement.

5.6. Communication : l'agent peut communiquer avec d'autres agents, Localement, ou distant

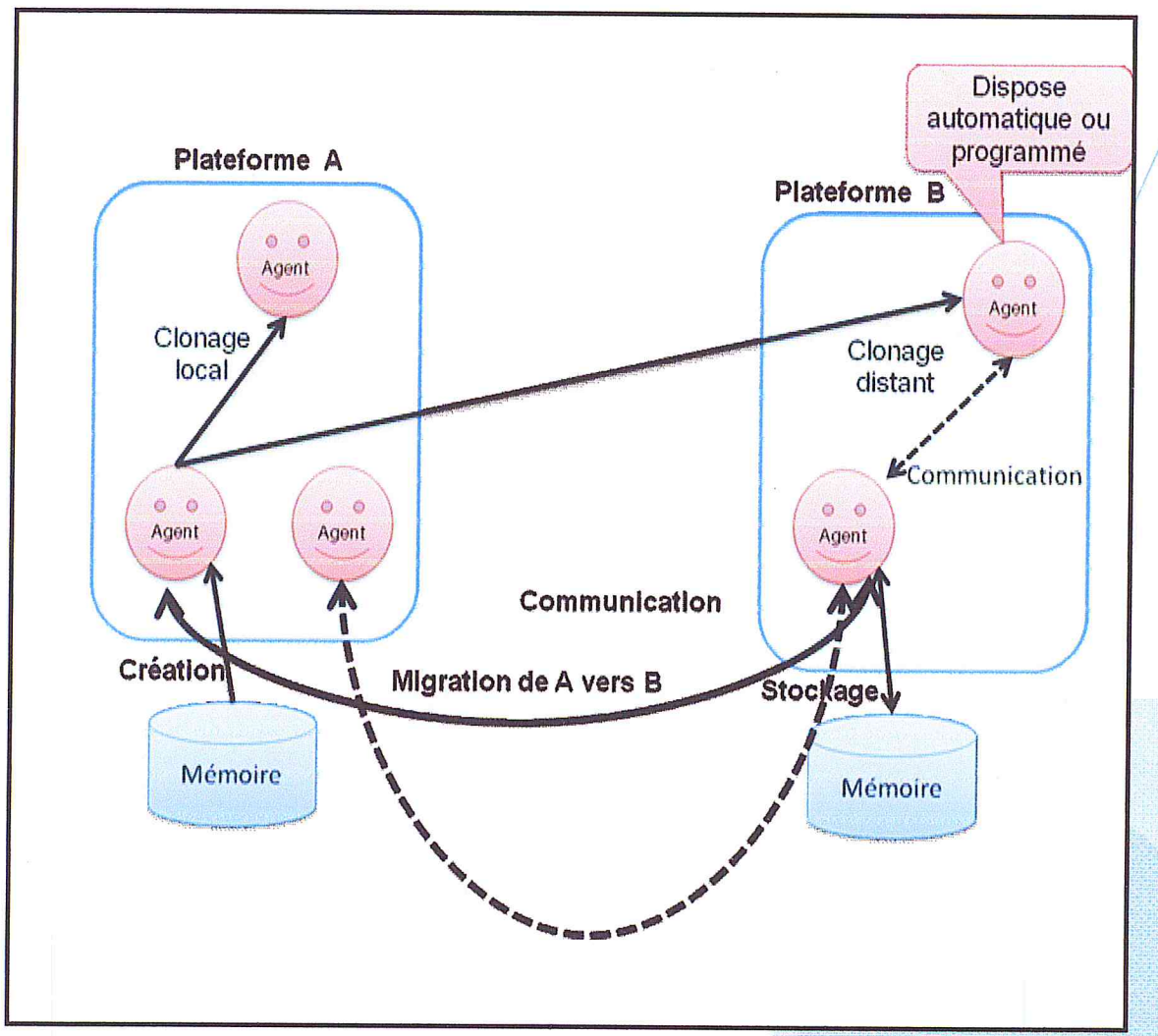


Figure 3.4 : cycle de vie d'un agent

6/ langage de programmation des AM [Mem02]

La notion d'agent mobile pose un problème d'intégration dans les langages. Le système doit gérer l'isolement de l'agent de manière à ce qu'il devienne autonome pour sa mission, les systèmes utilisant JAVA sont confrontés d'une part au problème d'isolation du code et des données de l'agent et d'autre part au problème de la migration des threads, ils résolvent le premier par la sérialisation : l'agent emmène avec lui une copie de tous les objets qu'il référence. Et ils contournent le deuxième en redémarrant l'agent avec un seul flot d'exécution sur une procédure d'entrée unique c'est-à-dire que l'état d'exécution n'est pas complètement restaurer.

7/ Points forts des AM [Mem02]

- **Personnalisation** : il est plus facile à un utilisateur de personnaliser son agent qu'un programme résident sur un serveur distant.
- **Survivabilité** : alors qu'un programme classique est lié à une machine, un agent mobile peut se déplacer pour éviter une erreur matérielle ou logicielle, ou tout simplement un arrêt de la machine.
- **Représentation d'un utilisateur déconnecté :** un agent peut continuer à parcourir le Web même sans interaction avec l'utilisateur, grâce à son autonomie.
- **Réduction de la charge du réseau** : en se rapprochant des données sur lesquelles il veut travaille, un agent mobile peut permettre de diminuer le nombre de messages de communication, et par la même, la charge du réseau.
- **Facilité de développement :** le concept d'agent devrait permettre de développer des applications mobiles plus facilement en masquant à l'utilisateur les problèmes liés au transport dans les réseaux.

Synthèse:

Les agents mobiles visent principalement des applications réparties sur des réseaux à grande distance, car ils permettent de déplacer l'exécution vers les serveurs et de diminuer ainsi le coût d'accès à ces serveurs. Néanmoins ces agents sont limités par certain problème dont la sécurité semble la plus sérieuse:

- **Manque d'application :** où les agents mobiles apportent un avantage certain.
- **Sécurité :** c'est une difficulté pour le développement de système fonctionnant en environnement ouvert (internet) et non fiable (réseaux mobiles).
- **Manque d'infrastructure et de standards :** aucun standard n'est réellement appliqué et leur conception varie encore beaucoup d'un système à l'autre malgré les efforts d'organismes normalisateurs comme la FIPA (foundation for intelligent physical agents). De même, les difficultés non résolues empêchent la formation d'une infrastructure suffisamment fournie pour permettre le développement d'applications économiquement intéressantes.

Conclusion:

Les agents mobiles semblent ouvrir une nouvelle voie pour la conception future D'applications réparties grâces aux multiples avantages qu'ils offrent notamment leurs Qualités asynchrones et leur utilisation modérée des ressources du réseau. Mais vu les faiblesses cité ci-dessus il semble indispensable aux agents d'être combiné a leur propre infrastructure et plateforme pour s'exécuter et interagir avec les autres applications.

Chapitre 3:

La recherche d'information

1. Introduction :

Il est devenu difficile de garantir aux utilisateurs d'internet un accès simple et rapide a l'information vue l'évolution exponentielle des contenu des sites en quantité et diversité.

Et pour cela une combinaison d'outils de recherche fiables aux SRI s'avère indispensable afin de pouvoir collecter des pages web et présenter les meilleurs résultats conformément aux demandes de l'utilisateur.

Cette partie va être abordée par les concepts de base des systèmes de recherche d'information ainsi cité leurs problèmes par la suite une évaluation de certains SRI par agent mobile existant afin d'extraire ses faiblesses. Et la fin on va conclure avec les problèmes qui les confrontent.

2. Concepts de base de recherche d'information :

2.1. Définition de la RI :

« Abrégée en RI ou IR (Information Retrieval) La Recherche d'information est un domaine de l'informatique qui s'intéresse à l'organisation, au stockage et à la sélection d'informations répondant aux besoins des utilisateurs. » [Bou04]

2.2. Définition d'un SRI:

Un système de recherche d'information SRI est un système qui permet aux utilisateurs l'accès à un ensemble de documents par leur contenu sémantique selon une demande d'information faite par l'utilisateur souvent exprimer sous forme de mots-clés (requête) et qui est faite sur un très grand nombre de document met en correspondance avec une représentation du contenu des documents au moyen d'une fonction de comparaison. [Jas00].

2.3. Définition du Moteur (ou engin) de recherche :

C'est un outil informatique de simple accès, mis à la disposition des utilisateurs pour chercher de l'information sur internet dont la localisation qui lui est inconnu ou disséminés sur le web. [Bae92]

3. Concepts de base d'un SRI :

Afin d'extraire l'information du web les SRI doivent s'entraider avec des outils de recherche tel que : des moteurs de recherche ou annuaire de recherche, qui permettent un accès à une information dont l'unité indivisible la plus petite est le document, cette combinaison doit avoir une certaine similarité dans le fonctionnement qui comprend cinq concept de base :

3.1/Une collection de documents:

Qui constitue l'ensemble des informations exploitables, compréhensibles et accessibles par l'utilisateur. Cette collection peut suivre quatre types de logique de recherche selon l'outil de recherche :

3.1.1/ La logique géographique :

Permet de trouver un ou plusieurs sites web par localisation géographique

3.1.2/ La logique thématique :

C'est une sorte de catalogue thématique quand peut consulter à l'aide d'une liste de mot clé

3.1.2/ Le logique index :

A travers d'une requête formulée par l'utilisateur de mots clé au prés d'un moteur de recherche qui retourne en revanche l'adresse du document ainsi qu'un résumé du contenu et puis l'utilisateur va faire une recherche manuel de ces besoins

3.1.3/ Le logique méta-index :

Ce sont des programmes informatiques qui réalisent une interface évoluée et rapide entre l'engin et l'internet

3.2/ Un besoin en information :

Se représente par un mot clé qui illustre un besoin ou une demande d'information faite par différents utilisateurs [Bae98] ces besoins se diffères aussi en :

3.2.1/ Besoin vérificatif:

Vérification d'une information qui connait déjà.

3.2.2/ besoin thématique connu:

Clarification d'une ambigüité ou a sa correction ou bien a la revoir dans un sujet et domaine connus.

3.2.3/ Besoin thématique inconnu :

Recherche de nouveaux concepts ou relations sur des sujets ou domaines qui lui sont totalement inconnu.

3.3/ Un processus de représentation du contenu des documents et des besoins en information de l'utilisateur, (indexation):

Les documents et requêtes récupérés sont structurer dans cette étape selon une description qui comporte les concepts les plus importent du document ou de la requête.

3.4/ Un processus d'appariement document-requête :

C'est la comparaison document-requête offre un moyen pour afficher les résultats de recherche dans un ordre de pertinence du meilleur au moins bon. Ce résultat peux être aboutis selon un calcule de pertinence qui est propre a chaque modèle de recherche d'information.

3.5/ Un mécanisme de reformulation de requêtes :

Cette étape consiste à reformuler une requête (après avoir eu le résultat et qui est jugé par l'utilisateur qui ne répond pas a sa demande) en ajoutant des mots clé consistant ou de mieux formuler chaque mot clé est a déjà utilisé.

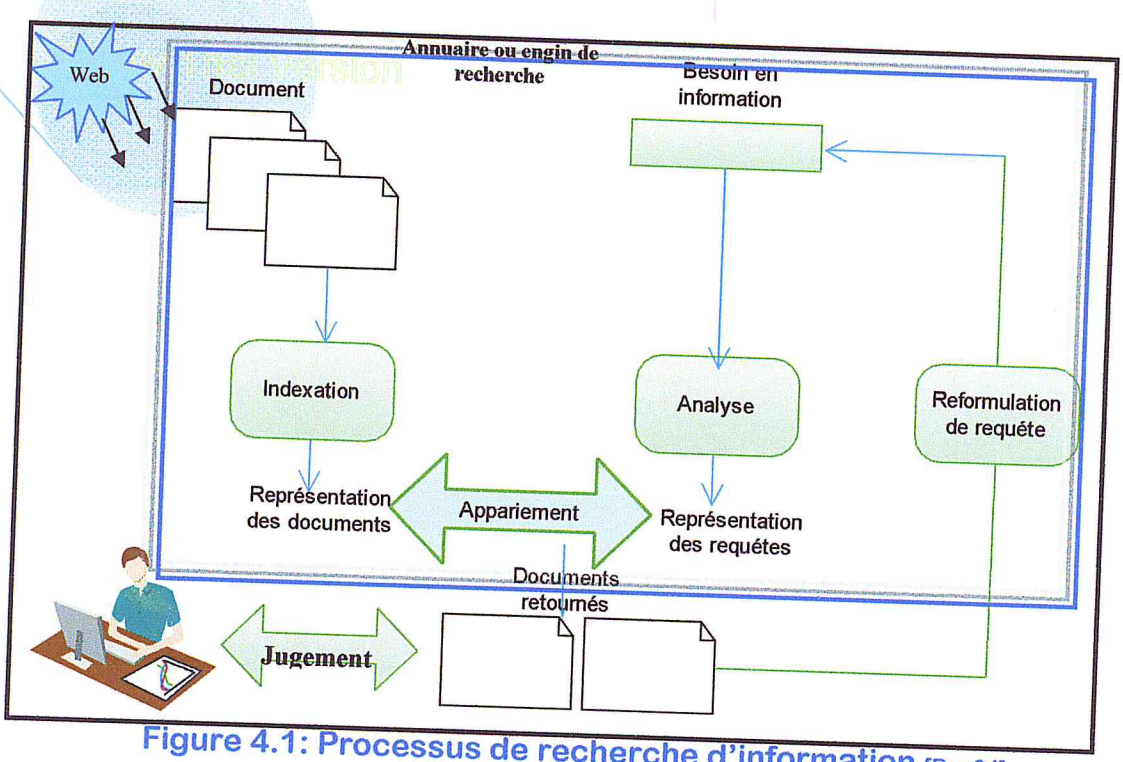


Figure 4.1: Processus de recherche d'information [Bou04]

4. Fonctionnement Générale d'un SRI :

Les systèmes de la recherche d'information peuvent être divisés en trois grandes phases selon leurs fonctionnements:

4.1/ L'indexation ou l'analyse:

Cette opération ce fait après la collection des documents qui consiste a les analysés et crée un ensemble de mot-clé ou un descripteur de document ou requête qui consiste a trouvé leurs concepts importent qui seront par la suit mis dans la base d'index selon des modèles (qui vont êtres cités plus bas) et utiliser par le système lors d'une recherche. Cela offre une efficacité et un cout acceptable

Types d'indexations :

L'indexation peut être faite de l'une des méthodes suivantes:

- **manuelle:** chaque document est analysé par un spécialiste du domaine ou par un documentaliste. Ce type donne les meilleurs résultats par rapport aux autres méthodes d'indexation. Mais vus l'effort en temps et en nombres de personnes qu'elle exige, et les conflits entres spécialistes lors de choix des termes qui peux êtres différents pour le même document donne un point faible a cette indexation.

- **automatique** : à l'aide d'un processus entièrement informatisé.

- **semi-automatique:** après avoir fait une indexation automatique, un spécialiste ou un documentaliste décide des termes plus significatifs .Ces indexeurs utilisent Un thésaurus ou une base terminologique, qui est une liste mots clés qui suivent des règles de terminologies propres et reliés entre eux par des relations sémantiques.

4.2/ L'appariement document-requête:

La comparaison entre le document et la requête revient à calculer un score, supposé représenter la pertinence du document vis-à-vis de la requête, cette évaluation est propre à chaque modèle.

La valeur est calculée à partir d'une fonction de similarité

RSV (Q,d) (Retrieval Status Value) **Q** est une requête et **d** : un document.

Cette mesure tient compte du poids des termes dans les documents, déterminé en fonction d'analyses statistiques et probabilistes.La fonction de similarité permet ensuite d'ordonner les documents renvoyés à l'utilisateur. Et comme l'utilisateur se contente des premiers documents renvoyer alors il considère le SRI comme mauvais s'il le document recherché ne se trouve pas parmi les premiers résultats (10 à 20 premiers résultats)

4.3/ La reformulation:

La requête initiale est vue en recherche d'information comme un moyen permettant d'initialiser le processus de sélection d'informations pertinentes .pour cela quelques systèmes incorporent une étape supplémentaire de reformulation de la requête afin de rapprocher au mieux la pertinence système de la pertinence utilisateur.

La reformulation de la requête consiste à modifier la requête de l'utilisateur par ajout de termes significatifs et/ou ré-estimation de leur poids

Types de reformulation :

Les termes peuvent provenir:

- **De documents jugés par l'utilisateur**. On parle alors dans ce cas de la réinjection

De pertinence, communément appelée Relevance feedback ou Retour de pertinence.

- **Des sources construites manuellement (thésaurus),** ou automatiquement à partir

Des documents de la collection.

5. Les modèles d'indexation:

Notons qu'il existe une très grande collection de documents qui se trouve sur tout le réseau Internet. Et l'étape de l'indexation permet une représentation des documents se trouvant dans cette collection selon un modèle utilisé en spécifiant une liste de mots clé {c0, c1, c2, .., cN} avec leurs fréquence d'apparition (ou poids) {f0, f1, f3, …, fN}. Il existe trois modelés de sauvegarde d'information :

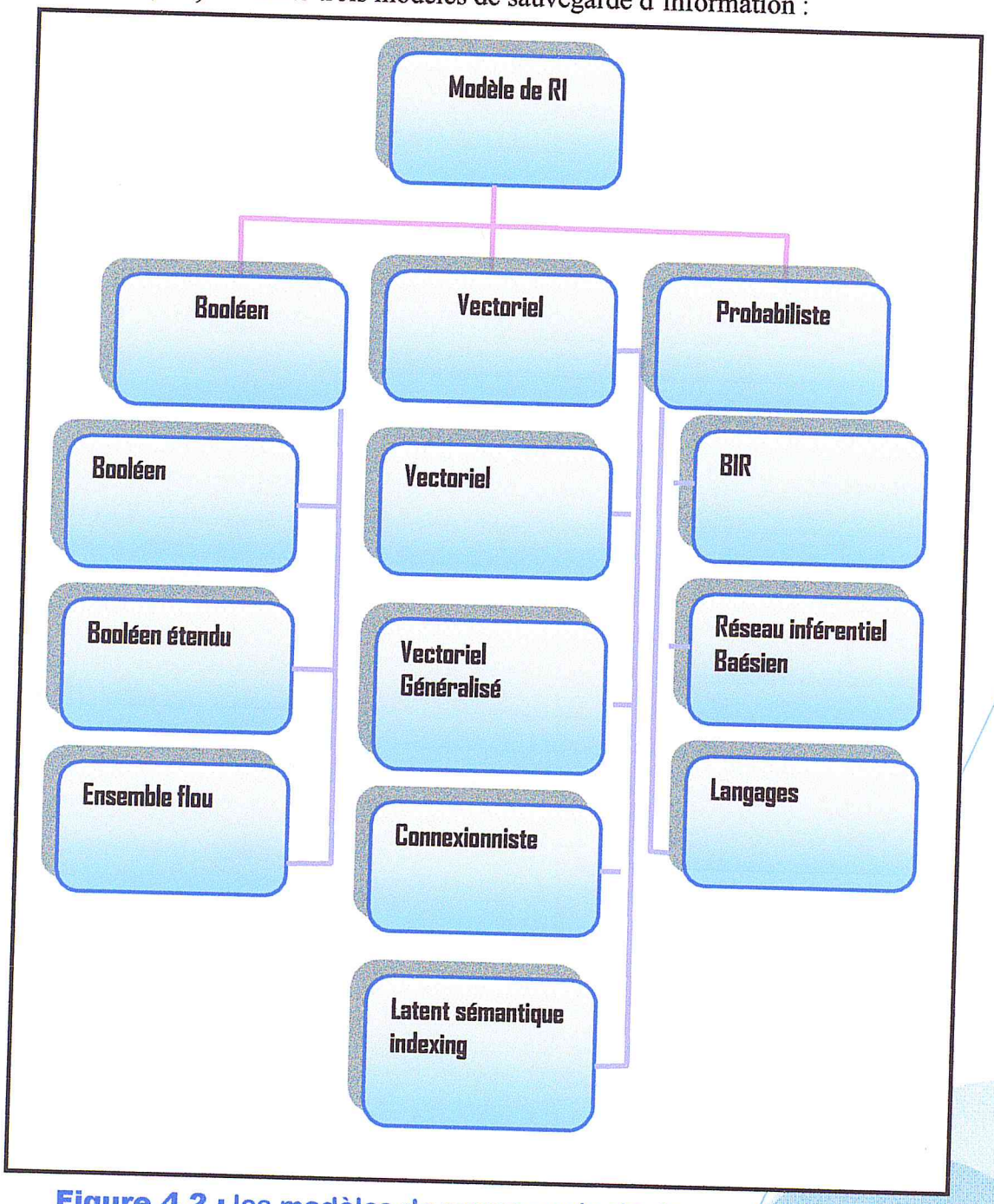


Figure 4.2 : les modèles de sauvegarde de document [Pic03]

5.1/ Les modèles basé sur la théorie des ensembles :

Qui traite une requête comme une expression logique car elle est exprimée par une liste de termes et des opérateurs logiques :

Conjonction (ET), disjonction (OU) et négation (NON)

On définit l'expression **R (D,t)** tel que **D** est un document et **t** un terme, comme étant vraie si le terme se trouve dans le document et comme fausse sinon.

Type : Il existe des modèles dérivés du modèle booléen comme

- **le modèle booléen étendu**
- **le modèle basé sur les ensembles flous (la logique floue)**

5.2/Les modèles algébriques :

Reposent sur la théorie algébrique. Le modèle le plus connu est le modèle vectoriel, La pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance dans un espace vectoriel.

En effet, pour chaque document dans un ensemble, on compte le nombre d'occurrences de chaque mot (fréquence d'occurrence du terme), notée **tf**.

Cette vérification des occurrences forme un vecteur pour chaque document et l'ensemble des vecteurs forme une matrice appelée matrice d'occurrences.

Au lieu de compter le nombre d'occurrences, on peut aussi simplement vérifier la présence du terme et attribuer la valeur 1 lorsque le terme est présent, et la valeur 0 sinon,

Il y a plusieurs autres facteurs dont on doit tenir compte pour garantir la pertinence de la recherche dans ce type de modèle :

- **Le volume du document :** plus le document est volumineux contiendra plus de termes, mais il n'est pas nécessairement toujours plus pertinents. Il faut donc « pénaliser » les documents très volumineux et aider l'utilisateur à trouver des documents courts qui répondent bien à la requête.
- Toutes choses étant égales, plus un terme est fréquent dans un document, plus le document est pertinent par rapport à ce terme.
- On doit attribuer plus d'importance, dans la requête de l'utilisateur, aux termes plus rares. Cette rareté est calculé : Un facteur **idf** qui sert à tenir compte de la rareté relative de certains termes tel que :

idf= log (|D|/df) ou **|D|** est le nombre de documents, et **df** le nombre de documents où un terme donné apparaît.Sur la matrice d'occurrence en multipliant les fréquences **(tf)** par le facteur **idf** correspondant.

Type : Plusieurs modèles s'inspirant du modèle vectoriel ont été proposés.

- **le modèle vectoriel généralisé,**
- **le modèle connexionniste**
- **le modèle LSI (Latent Semantic Indexing)**

5.3/ les modèles probabilistes :

Se basent sur la théorie des probabilités. La pertinence d'un document vis-à-vis d'une requête est vue comme une probabilité de pertinence document/requête.

Type :

- **le modèle BIR (Binary Independence Retrieval).**
- **le modèle inférentiel bayésien**
- **le modèle de langage.**

6. Architecture générale des moteurs de recherche:

Afin de procurer de l'information une simple opération doit être réaliser qui consiste à donner au moteur les mots importants concernant le sujet des informations recherchées, dits mots-clés, ou la description même du document recherché, et ce moteur a pour effet de retourner comme résultat de ses recherches la liste de toutes les pages web relatives à ces mots-clés, ou bien celles correspondant à la description passée en paramètre. La description de ce processus est présentée si dessous :

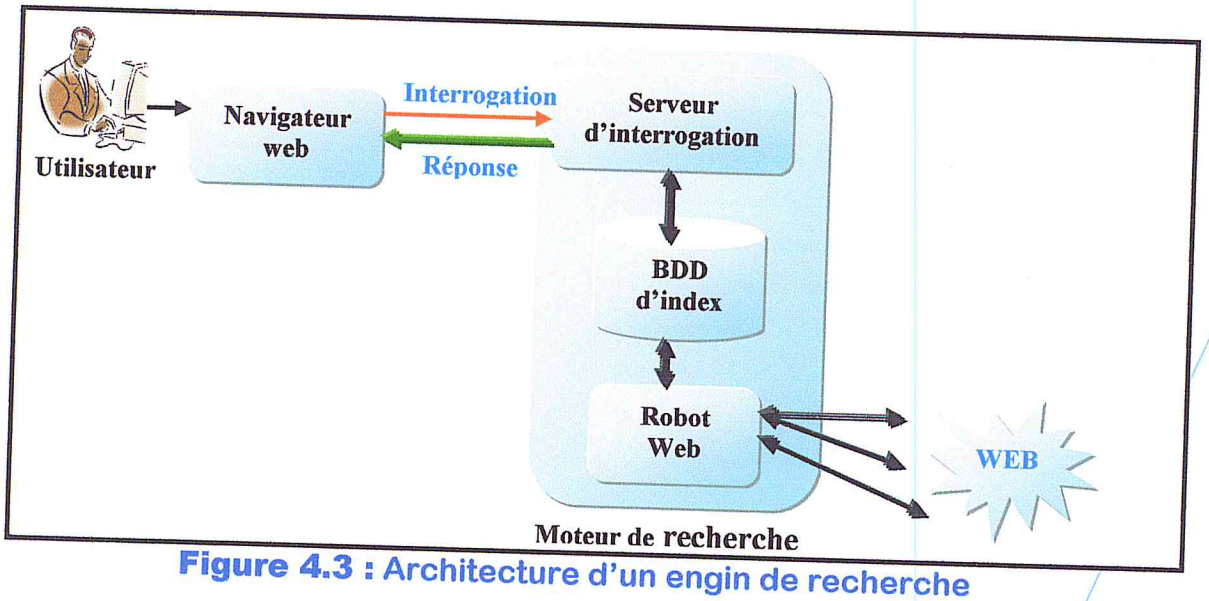


Figure 4.3 : Architecture d'un engin de recherche

Notons tous d'abord qu'un moteur de recherche est composé de plusieurs éléments soient :

6.1/ Un robot :

Un programme dont l'unique fonction est l'exploration à fréquence régulière des liens hypertextes et parcours les différents ressources (serveurs web, forums,..) à partir d'un ensemble donné de pages initiales pour archiver le contenu des sites qu'elle contient. Effectuées par le module de collecte dit : spider, crawler, Worm ou search engine.

Ses ressources peuvent être réellement indexée sont appeler aussi web invisible car il comprend seulement 1/500 ième des documents accessibles sur le web. En effet les spiders prennent en compte :

- Les documents de formats (PDF, Doc, xls…..)
- Les sites d'une certains profondeurs

6.2/ Un moteur d'indexation :

C'est l'étape qui suit la scrutation, car chaque page trouvé serra associer a un terme selon plusieurs critères de choix :

- Choix selon les termes fréquemment utilisé sur le document
- Choix selon les termes du titre ou d'autres parties significatives de la page.

Ces index seront considérer par la suite comme la base de données du moteur de recherche. Ils contiennent un index principal, qui liste les milliards de pages capturées par le crawler, ainsi qu'un index de fichiers inverses, contenant les termes d'accès aux pages web, comme les mots-clés et les descriptions textuelles des documents indexés.

Cette indexation peut effectuer l'un des trois types d'analyses :

- **Morphologique :** Scanner de simples mots avec une gestion limitée de la casse et des mots vides (suppression).
- **Lexicale** ou **lemmatisation :** qui prend les mots mais non conjuguées, et a la première personne du singulier, cette analyse a une faible utilisation.
- **Syntaxique :** c'est une analyse des expressions complètes, qui est le plus souvent appliquée aux formulaires servant à la recherche dans la base de données.

6.3/ Un moteur d'interrogation :

Effectuer par le module de recherche, combine les fonctionnalités entres les utilisateurs et la base de données crée par le moteur d'indexation.

Ce lien se fait par une interface de requête par mots clés présenter sous forme d'un formulaire de saisie mis à la disposition des utilisateurs des moteurs. De cela on peut certifier que se module a pour rôle de :

- faire un appariement entre les termes de la requête avec le contenu de la base de données du moteur de recherche.
- Afficher les titres ou les URL correspondants aux pages coïncidentes.

7. Propriétés des moteurs de recherche :

Le développement d'Internet exige un remplacement des moyens de communication et d'information actuel tout en mettons les ressources en communs et d'y accédés.

Dans cette état de l'art nous avons choisis de décrire les moteurs de recherche vue leurs large utilisation qui est due a leurs fonctionnalités :

- La recherche d'information qui a pour objectif : trouvé l'information désirée pertinente en toute rapidité.
- Référencier les sites web afin de permettre la fourniture d'informations par les sociétés de services ainsi que les webmasters privés. Pour :

1. permettre aux utilisateurs particuliers d'accédés aux données désirer: utilisation à but informatif, tels les journaux Online,
2. la diffusion non sollicitée de données imposées aux utilisateurs d'un Moteur: utilisation à but publicitaire, qui est mise en œuvre grâce à un détournement des critères.

- La domination de certains moteurs qui est due a la possibilité offerte aux utilisateurs de s'en servir avec n'importe quelle langue. exemple le moteur : Google.
- Les avantages dans le domaine de la recherche allant de leur mode d'indexation des documents disponibles sur le web jusqu'à leur style d'utilisation par leurs clients, par rapport aux répertoires thématiques :

	Moteurs de Recherche	Répertoires Thématiques
Type de Recherche	Recherche par **interrogation** pour une recherche pointue	Recherche par **navigation** pour une recherche générale
Alimentation des bases de données	Par des **robots** logiciels qui scrutent les pages web et les emmagasinent Ils constituent des index ou bases de données	Par des **êtres humains** qui organisent les sites visités en répertoires Ces experts classent les sites en diverses catégories
Types d'engins	Moteurs ou métamoteurs Langues indifférentes	Généraux ou spécialisés Une langue par répertoire
Déroulement des recherches	Requête par mots-clés Recherche transparente	Sélection par thèmes Liste de liens de sites
Affichage des résultats	Mots-clés en gras suivis d'une description de la page web correspondante	Identique à une table des matières dans un livre

Tableau 4.1: Comparaison entre moteur et annuaire de recherche

8. Critères de pertinences des SRI :

On peut juger de la pertinence d'un SRI de la pertinence de la recherche de son outils .Cela à travers plusieurs critères qui sont liées a l'algorithme de recherche vue qu'il a pour rôle de scruter la base de données en fonction des priorité affecter a chaque critère.les concepteur des moteurs de recherche donnent un choix de priorités de pertinence qui va par la suite juger de l'orientation de cet outil ou bien de l'utilité du moteur. L'évaluation des systèmes peut être abordée selon, deux aspects :

1. **un aspect efficience :** regroupe le temps et l'espace : plus le temps de réponse est court et plus l'espace occupé par le système est faible, meilleur est considéré le système.
2. **un aspect efficacité :** concerne la capacité du système à sélectionner le maximum de documents pertinents et un minimum de documents non pertinents, ces deux opérations évaluées par des mesures de précision et de rappel que nous définissons ci-après. Voire aussi :
3. **l'indice de popularité**: le moteur prend en compte la popularité du site hébergeant le document, qui est définis par :
    1. le nombre de liens pointant vers ce site.
    2. la popularité même des sites proposant ces liens.

Pour Google le facteur de popularité est effectué par le calcule du Pagerank d'un document, on utilise la formule suivante:

$$PR(A) = (1-d) + d(PR(TI)/c(T1) + ...+PR(Tn)/C(Tn))$$

- **Ti** : les documents pointant sur le document **A** cible.
- **C(Ti)** : le nombre de liens de chaque document **Ti**.
- **d ou damping factor:** Compris entre 0 est 1 : Représentant la probabilité que le document puisse être trouvé non conforme aux demandes d'un utilisateur. Constant égal à 0.85.

4. Le temps de réponse acceptable :

Un SRI doit pouvoir fournir à l'utilisateur les documents correspondants à sa demande dans des temps très courts.

5. La présentation des résultats claire avec facilité d'utilisation :

Capacité du système à Comprendre les besoins de l'utilisateur et à mettre en valeur les documents correspondants a ceux-ci. Ceci est lié à l'interface avec l'utilisateur.

6. Le nombre total de documents pertinents retournés :

Ces mesures permettant d'évaluer la performance globale du système en fonction ou non du nombre de documents pertinents total.

7. Le rang du premier document pertinent :

Cette mesure a été proposée pour prendre en compte la satisfaction de l'utilisateur qui chercherait un seul document pertinent (comme c'est éventuellement le cas pour les moteurs de recherche sur Internet).

8. La longueur de recherche :

Elle est égale au nombre de documents non pertinents que doit lire l'utilisateur pour avoir un certain nombre n de documents pertinents.

9. Calcul de pertinence:

Soient |A| le nombre de documents renvoyés par un système pour une requête donnée,

Et |R| le nombre de documents pertinents dans la collection pour cette requête

Et |Ra| le nombre de documents pertinents renvoyés par le système selon la figure suivante :

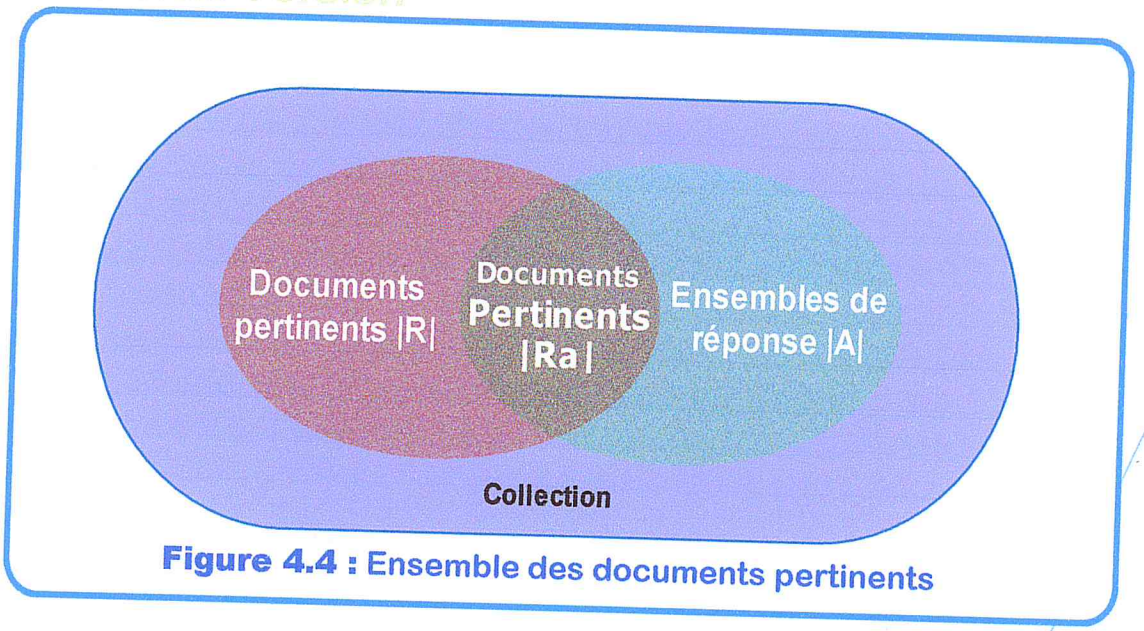


Figure 4.4 : Ensemble des documents pertinents

- **Précision :**

La précision mesure la proportion de documents pertinents relativement à l'ensemble des documents restitués par le système.

Elle est exprimée par : Précision = |Ra| /|A|

- **Rappel :**

Le rappel mesure la proportion de documents pertinents restitués par le système relativement à l'ensemble des documents pertinents contenus dans la base documentaire.

Il est exprimé par : Rappel = |Ra|/|R|

10. Les problèmes des SRI :

Comme toute autre technologie la recherche d'information rencontre plusieurs problèmes, citant:

- le fait que le web s'élargit de jour en jour, la création d'une collection de document statique est chose quasi impossible. Même en utilisant des robots explorateurs, on ne peut jamais garantir une couverture totale du web.
- les SRI retourne en généralement une quantité importante de documents ce qui provoque la perte des documents pertinents dans une très grande quantité de document non pertinents
- les SRI actuels sont conçu pour faire de la recherche dans des documents textuels .la présence des documents non textuel nécessite des nouveaux mécanismes de recherche et d'indexation.
- La recherche multi-langue n'est pas possible que dans quelque SRI. La recherche d'un mot clé dans une langue retourne toujours des résultats dans la même langue (sauf si le mot clé s'écrit de la même manière dans plusieurs langues), Et l'inefficacité de certains méthodes de génération de mots clé qui ne présent pas réellement le contexte du document.
- Les SRI nécessite une connexion permanente durant toute l'opération de la RI. n'importe quelle déconnexion avant la réception des documents engendre une réinitialisation de l'opération de la recherche.

Exemples de SRI par agent mobile

1. Introduction:

La recherche d'information cour différents types de problèmes qui peuvent être résolue par la structure et le comportement d'un agent mobile, néomoins dans la partie qui suit nous allons présenter quelque exemple de SRI basé sur les agents mobiles:

2. Exemples de SRI par agent mobile:

2.1. Le système Calvin [Cal02] : C'est un système de recherche d'information basé sur les agents qui se compose de :

1 .La Geneva : le middleware, c'est la couche qui fournie le service de base pour les SMA, tel que : la communication d'agents, l'authentification, et l'encryptage, elle utilise des spécifications de open XML pour permettre des multiples types d'agents et une range de tache. De plus cette couche permet a un certain type de d'agent de fonctionner tel que :

- **MatchMaker** : c'est un agent qui fournit l'authentification et coordonne entre les agents.
- **DataManager** : Fournit le stockage persistant protégé par mot de passe à d'autres agents.
- **LogBot** : Fournit une notation centralisée à laquelle tous les agents sur le réseau peuvent écrire.Chacun de ces agents fournit les facilités générales pour permettre aux autres agents de fonctionner.

2. le Calvin : c'est le regroupement d'un ensemble d'agent qui permettent l'information, la récupération et la spécification, a ce niveau chaqu'un des agents a un rôle spécifique :

- **L'agent d'Interface** (*Calvin Web*) **:** Fournit la communication bi-directionnelle entre le SMA et un browser de Web d'utilisateur.
- **Agents d'Analyse** : Ces agents analysent le comportement d'utilisateur et développent un profil d'utilisateur reflétant les intérêts de l'utilisateur, ils se devisent en :

1. *TFIDF Agent* : cette agent se renseigne sur les intérêts des utilisateurs en appliquant l'algorithme d'analyse des textes « Term Frequency/Inverse Document Frequency.
2. *WordSieve Agent* : Applique l'algorithme « WordSieve » pour développer un profil d'utilisateur et se renseigner sur l'intérêt courant de l'utilisateur.
3. *DocStats* : Accumule des informations statistiques sur les documents qu'un utilisateur a consultés.

Agents de recherche : Ces agents exécutent des recherches de fond pour l'utilisateur basé sur leurs profils d'utilisateur.

1. *AltaBot* : Exécute des recherches de fond sur Alta Vista.
2. *GoogleBot* : Exécute des recherches de fond sur Google

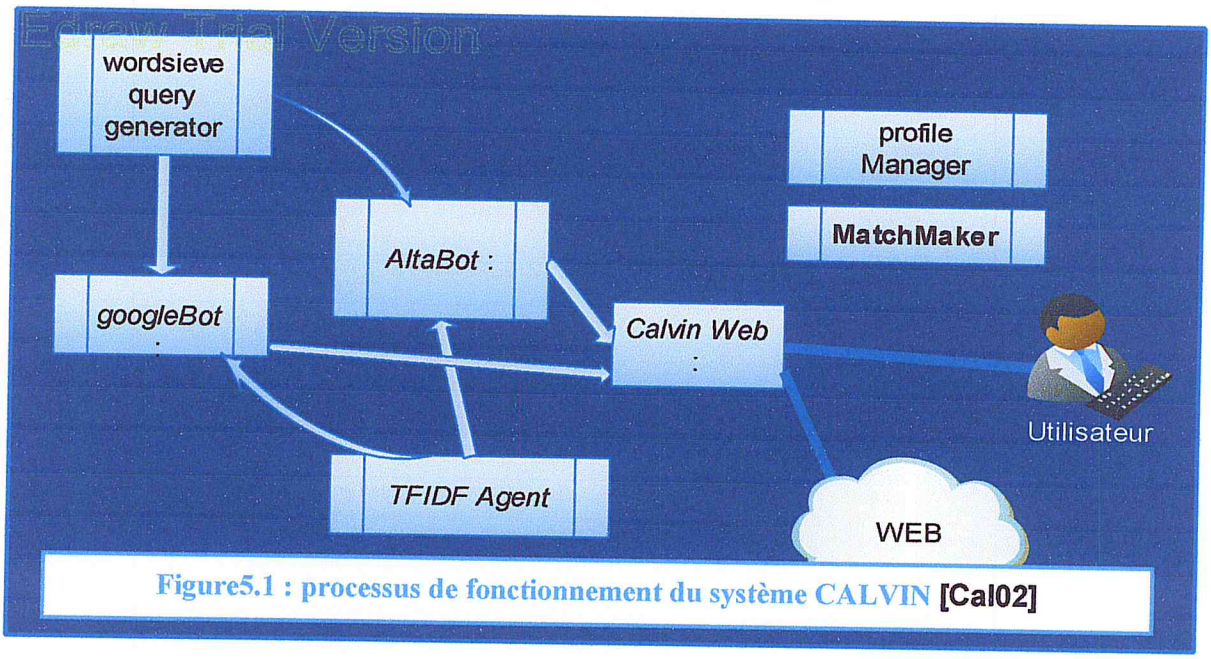


Figure5.1 : processus de fonctionnement du système CALVIN [Cal02]

2.2. NetSA [Tro06] [Jar10] :

C'est une architecture de système multi-agents pour la recherche d'information dans des sources hétérogènes et réparties. NetSA comporte différents types d'agents :

- **Des agents utilisateurs :** L'agent utilisateur, situé dans l'unité de communication avec l'utilisateur, est la porte d'entrée des requêtes externes au système. Ils sont en charge de la cueillette et du filtrage des informations provenant et allant vers l'usager
- **Un à plusieurs agents courtiers** : L'agent courtier, situé dans l'unité de traitement de l'information, associe aux différentes requêtes les agents qui sont capables d'y répondre
- **Un à plusieurs serveurs d'ontologie :** Un agent ontologie, situé dans l'unité de traitement de l'information, est en charge du maintien de la cohérence des concepts utilisés par les agents.
- **Un à plusieurs agents d'exécution :** Un agent d'exécution est en charge de la décomposition des tâches et du suivi du déroulement d'exécution des différentes sous-tâches
- **Des agents ressources internes ou externes :** Des agents ressources reliés chacun à une ressource d'informations et pouvant rapatrier et mettre à jour les données.

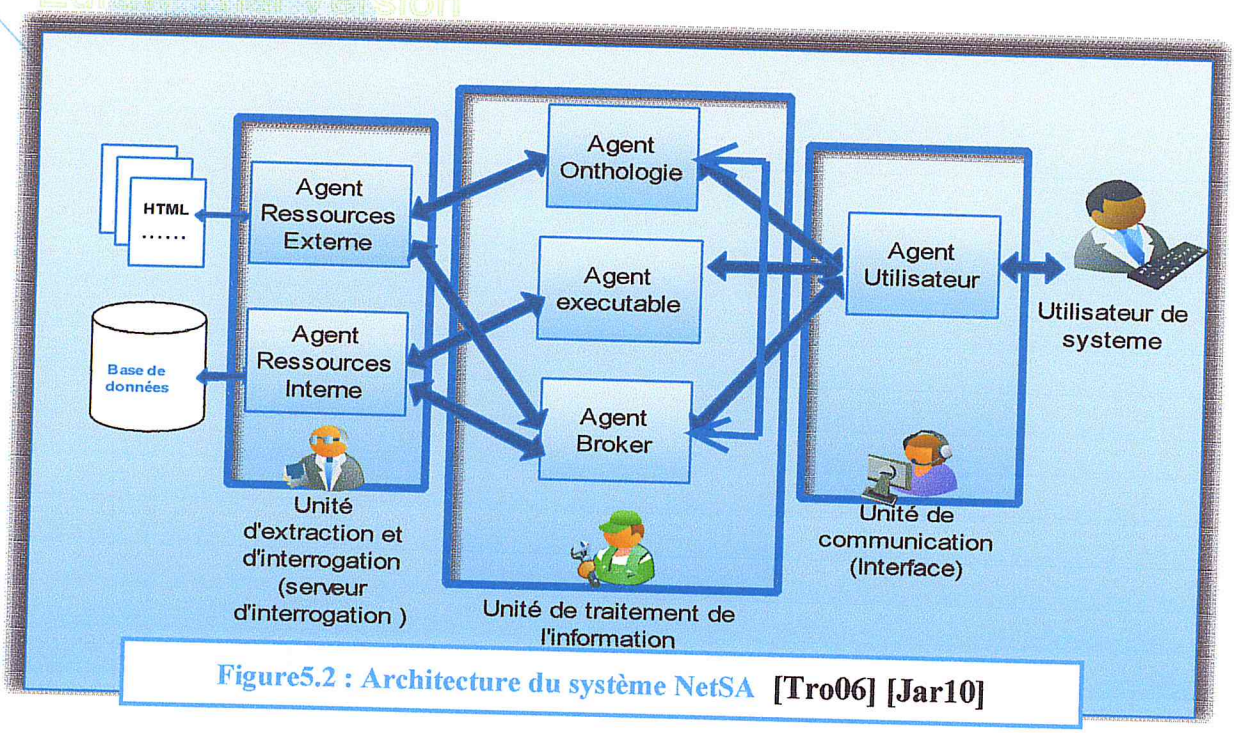


Figure5.2 : Architecture du système NetSA [Tro06] [Jar10]

2.3. The ZUNODL: [Fis79] [Feg01]

Le système de ZUNO Digital Library[1] est un système multi-agents qui permet à un utilisateur d'obtenir des données simple et organiser de la source incohérents et désorganisés tels que le Web. Trois types d'agents de ZUNODL sont :

1- Consommateur : représente des utilisateurs du système :

- **User Interface Agents (UIA)**: cet agent représente les besoin de l'utilisateur dans le système qui lui serra affecté afin de cherche de l'information sur la bibliothèque.

2- Producteur : représente les fournisseurs de contenu qui possèdent l'information que les clients consomment

- **Library Service Agents (LSA)** : C'est des agents représentant la bibliothèque numérique dans le réseau du propriétaire de l'information
- **Catalogue Agent (CA)** : Ces composants représentent les intérêts du propriétaire de l'information dans la bibliothèque numérique lui-même.

3- Facilitant : Organiser la communication entre les consommateurs et les producteurs

- **Search Agents (SA)**: Fournissez la fonctionnalité de base de moteur de recherche qui soutient ZUNODL. De plus ils fournissent des fonctionnalités plus avancé car

[1] La bibliothèque numérique est une collection de données ainsi que des services, qui est organisée et gérée de manière à aider l'utilisateur à atteindre ces données [Feg01].

il peut concaténée des modèles de représentation de l'information. Par exemple [Fis79], ils peuvent se servir des thesaurus et des taxonomies afin de manipuler des requêtes pour obtenir de meilleurs résultats pour utilisateur.

Synthèse :

Problèmes des SRI par AM :

Les SRI par agent mobiles courent de grands problèmes causés par la structure ou le fonctionnement des entités du SRI tel que les agents les plateformes ou l'architecture du systeme cela peux causer un problème de :

- Falsification de l'information.
- Destruction des agents.
- Déni de service.
- Ruine du système.

On va cité dans cette partie des exemples de menaces de sécurité causé sur les systèmes de recherche d'information par agent mobile, même si ces problèmes sont présent sur les systèmes de client serveur mais l'utilisation des agents mobiles permet une grande utilisation et exploitation

1.1. Attaque sur les systèmes d'agent :

un agent est une cible d'attaque vue son déplacement sur des milliers de sites qui peuvent être non sécurisé et aussi le dépassement sur le réseau qui est souvent non sécurisé de plus la communication avec d'autre agent qui peuvent détruire ou modifier son contenu, le site visiter présente un major problème vue qu'il peux le manipuler ou le détruire de cela il existe deux types de menaces:

- **Attaque de l'agent visiteur :**
  Il s'agit de l'attaque contre l'agent par des manipulations d'un hôte malicieux.
- **Attaque de l'hôte visité :**
  Il s'agit des menaces sur les données d'une machine hôte qui peuvent être causé contre un programme visiteur étranger dont ni les intentions réelle, ni la crédibilité ne sont encore établie.

1.2. Attaque de l'infrastructure de communication :

Le transfert des données précieuses par un agent dans une infrastructure de communication hétérogène requiert un minimum de garantie contre les attaques, les modifications et les ruptures brutales de liaison. Dans ce cas ces attaques sont lors du déplacement des agents.

1.3. Attaque entre deux agents :

Il s'agit des attaque qu'un agent peux subir de la part d'un autre agent on distingue :

- **Les attaques passives** : qui ne modifient pas le code de l'agent ainsi que les données qu'il transporte tel que l'espionnage des données transportées.
- **Les attaques actives** : qui consiste a l'altération du code des données de l'agent, par exemple la modification des valeurs des variables, ou toute modification du code.

2. Protection :

Il n'existe pas une technique universel afin d'assurer la sécurité des agents contre les objets malveillants mais on peut distinguer certain types de travaux qui on pour but d'amélioré les systèmes de recherche d'information par agent mobile

2.1. Protection d'un site contre un agent :

En ayant un accès direct à un site, l'agent mobile peut lire des informations et donc violer sa confidentialité comme il peut écrire et donc violer l'intégrité du site. Ce problème est aujourd'hui plus ou moins maîtrisé. En effet, plusieurs solutions ont été proposées comme :

- **Le back à sable (sendBoxing):**

Cette technique appeler aussi isolation consiste a attribué a l'agent visiteur suspecter d'être malveillant un adressage virtuel qui l'isole du reste de la plateforme pour ne pas influencer l'exécution du système.

- **La signature et l'analyse du code de l'agent :**

Accordé par son créateur consiste lui accordé une signature ou une autorisation de signature pour décidé si l'agent peux accéder au site.

- **Le contrôle d'accès (proof carying cod):**

L'agent doit être agrée par son auditeur qu'il suit les mesure de sécurité de sa plateforme afin de lui permettre d'atteindre le site cible.

2.2. Protection d'un agent contre un site :

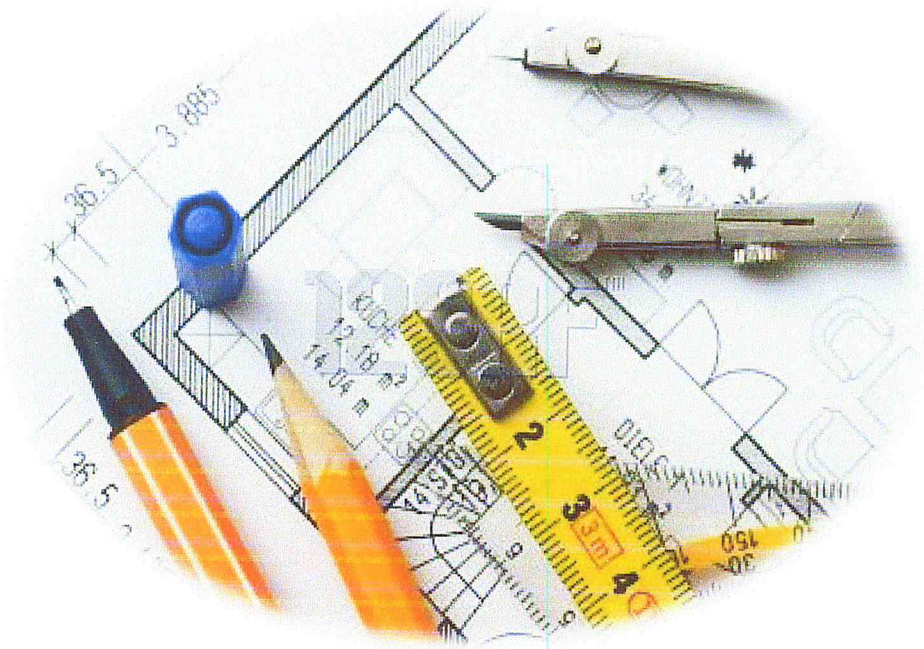
Lors de son exécution sur un site malveillant, ses éléments peuvent être une cible d'attaque. En effet, un site malveillant peut violer ses éléments. Contrairement à la précédente, la protection d'un agent contre un site ne dispose pas d'une solution.

- En interdisant à un agent mobile de migrer directement vers un site et à un site d'accueillir des agents, les agents mobiles peuvent être utilisés sans risques tout en gardant leurs performances.
- Ou on accorde des clés aux agents en vois de migration que seul si la clé privé correspond a la clé public le contenu de l'agent peux être décrypté.

Conclusion :

L'approche des agents mobiles est la meilleur technologie qui peut nous donné de bons résultats vis-à-vis d'autre technologie, nous pouvons dire que le problème le plus important empêchant l'adoption actuelles des agents mobiles est la sécurité, malgré la protection des sites qui est quasiment assurée, celle des agents n'a pas encore une solution.

En effet, les problèmes de sécurité des agents mobiles ne peux pas être toute a fait résolu car l'accord de code aux agents mobiles est un concept difficile a réalisé. Vu que la clé publique doit être comparer a la clé privé donc l'agent doit ajouter une nouvelle dimension à la programmation des agents mobiles et c'est le résonnement



Partie II :

Chapitre 4:

Analyse et conception

I. Description du système :

1. Introduction

Comme on l'a déjà vu précédemment, un système de recherche d'information SRI est un système qui permet l'accès à un ensemble de documents par leur contenu sémantique .cet accès est le résultat d'une recherche documentaire .celle ci consiste à mettre en correspondance une représentation du besoin de l'utilisateur (requête) avec une représentation du contenu des documents (fiches et enregistrements) au moyen d'une fonction de comparaison.

Malgré tous les progrès qu'a connus le domaine de la recherche d'information, La RI souffre encore de plusieurs problèmes, citant par exemples:

- Les SRI nécessitent une connexion permanente durant toute l'opération de la RI. Toutes déconnexions avant la réception des documents engendrent une réinitialisation de l'opération de la recherche.
- les SRI retourne généralement une quantité importante de documents ce qui provoque la perte des documents pertinents dans une très grande quantité de document non pertinents.
- le fait que le web s'élargit du jour en jour, la création d'une collection de document statique est chose quasi impossible. Même en utilisant des robots explorateurs, on ne peut jamais garantir une couverture totale du web.
- Les résultats des recherches dans les SRI sont souvent transférer en clair cela sous entend la non confidentialité de ces résultats et la possibilité de les falsifier.

La question qui se pose est : est ce qu'on peut régler au moins quelques uns des problèmes rencontrés dans les SRI moyennant la technologie d'agents mobiles? c'est ce qu'on va essayer de montrer dans la suite de cette partie a travers l'architecture SB **[Menacer, Drias , Sibertin Blanc, IADIS 2009]** (basé sur le modèle seller-buyer), on l'adaptant a la recherche d'information.

2. Utilité des AM dans les SRI :

Pourquoi utilisé les agents mobiles lors de la conception des systèmes réparties pour la RI ?

Il existe plusieurs arguments qui justifier l'utilisation des agents mobiles, on peut citer par exemples :

- Les agents mobiles supportent les opérations en mode déconnecté et les opérations de type asynchrone ce qui n'est pas le cas toujours avec les SRI à base du modèle client serveur classique.
- Les agents mobiles facilitent la personnalisation des services de la RI sur le réseau, ce qui apporte une plus grande flexibilité par rapport aux SRI à base du modèle client-serveur classique.
- Les agents mobiles réduisent le trafique sur le réseau lorsque la quantité d'informations délivrée par les SRI est très importante.

3. Limites d'utilisation des AM dans SRI:

D'un autre coté une autre question qui se pose aussi souvent : pourquoi ne pas utiliser (éviter) les agents mobiles dans les SRI ? .

La technologie des agents mobiles comme toute autre technologie est loin d'être parfaite, elle souffre encore de plusieurs problèmes citant par exemple :

- Chaque constructeur propose ces propres approches, ce qui rend les tentatives de normalisations difficiles.
- La portabilité partielle des agents diminue considérablement les performances de cette technologie.
- L'interopérabilité non assurée entre les différents systèmes d'agents mobiles qui existent, même si des grands efforts sont faits dans ce sens.
- Sans oublier l'aspect sécurité, le problème le plus important de toute cette technologie.

Comme on la déjà vu dans la partie des types de problème de sécurité sur les SRI par AM, le problème de la protection des hôtes peut être résolu dans certains cas avec certaines mesures. Cependant, il reste le problème de la protection des agents :

- Contre les attaques des hôtes malveillantes.
- Contre les attaques des autres agents (malveillants).

Et ces deux problèmes peuvent être résolus au moins partiellement par l'utilisation des mécanismes de protection à base de *PKI*.

Mais si on pense d'une manière un peu différente, et on va tenter de supprimer les deux problèmes (ce qui cause le problème) au lieu de chercher à trouver des solutions

partielles (qui ne répondent pas complètement aux exigences des systèmes à base d'agents mobiles) ?

L'idée et d'interdire aux agents de migrer directement vers les hôtes, et d'interdire aux hôtes de recevoir des agents, toute en assurant les communications entre les deux facteurs du système.

L'idée consiste à utiliser des sites intermédiaires qui vont accueillir les agents au lieu des hôtes, et les communications vont se faire entre ces sites qui vont représenter les agents (site proxy) et les hôtes.

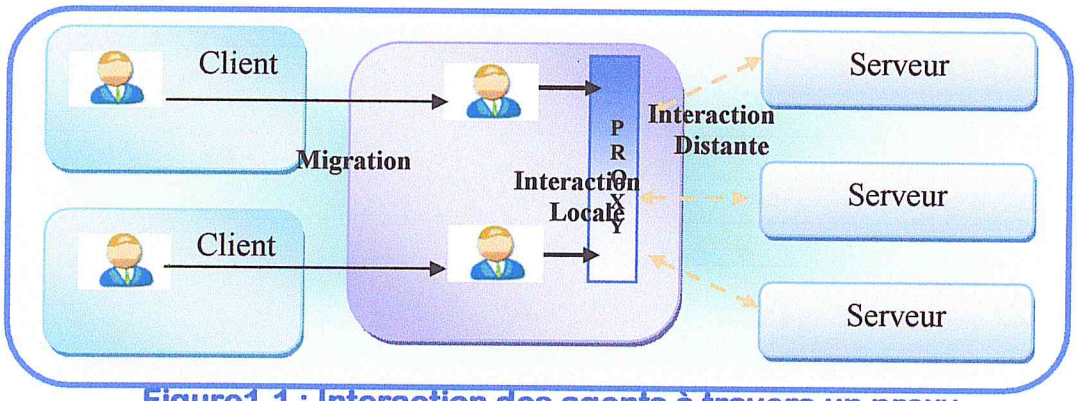


Figure1.1 : Interaction des agents à travers un proxy

Ainsi, les efforts de la protection des hôtes vont être orientés vers la protection des sites proxy ce qui est plus simple à réaliser. Mais cette solution va engendrer un autre problème : le fait que la communication entre les sites proxy et les hôtes se fait par le modèle client-serveur classique va limiter considérablement les points forts des agents mobiles.

Une amélioration de cette solution consiste à déléguer des agents mobiles sur les sites proxy qui vont représenter les hôtes et leurs services, ce qui va transformer les sites de proxy de simples sites qui reçoivent les agents mobiles (qui cherchent un service) à un lieu de rencontre (lieu de rendez-vous) pour les agents des deux types. Ce qui nous ramène à voir les sites proxy comme des **places de marché** ou les **agents acheteurs** (qui cherchent les services) et **les agents vendeurs** (qui fournissent ces services) se rencontrent et négocient.

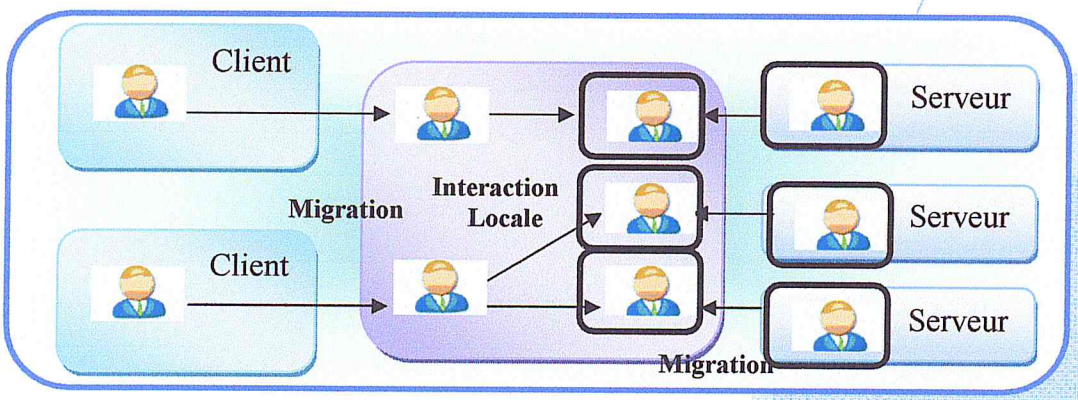


Figure1.2: Interaction des agents à travers une place de marché

4 .Prototype de l'architecture du modèle Seller-Buyer

On peut imaginer un système à base de place de marché (Market Place ou MP), qui va avoir l'architecture suivante :

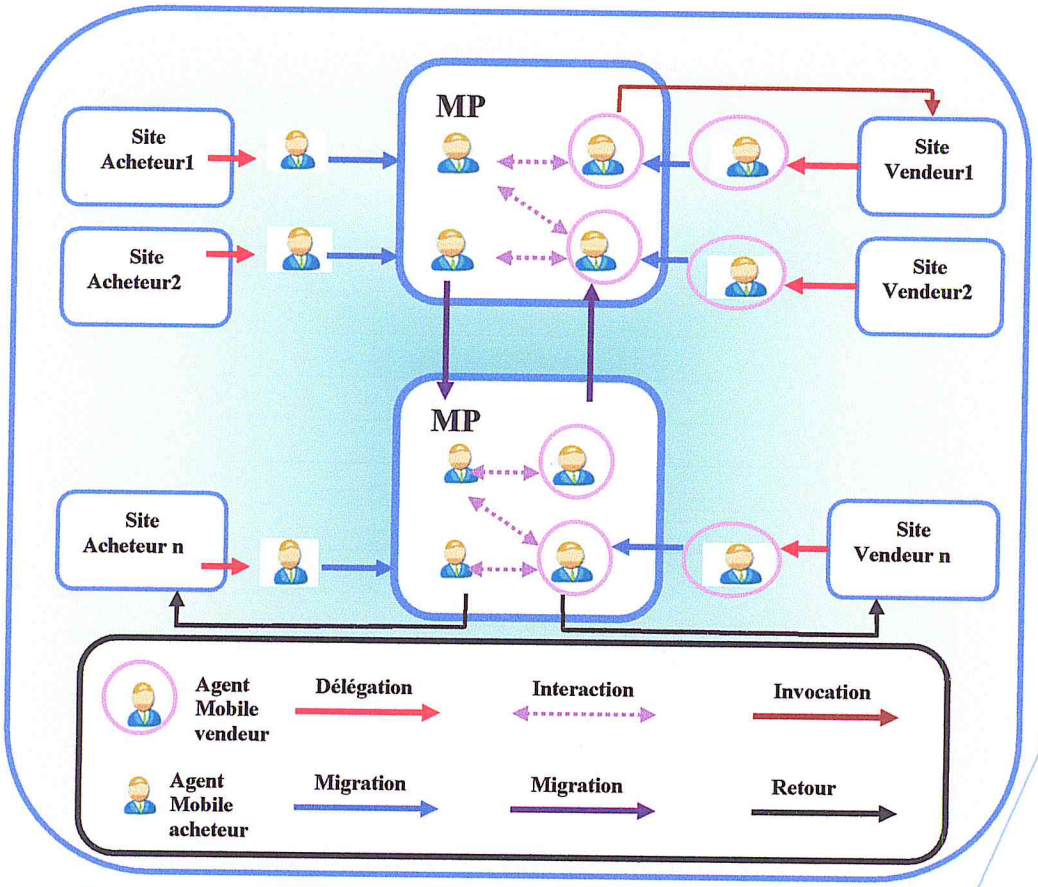


Figure 1.3 : Architecture Globale du modèle buyer seller

Pour faciliter les rencontres entre les agents acheteurs et les agents vendeurs, nous allons utiliser un troisième type d'agent qu'on appellera **agent facilitateur** qui va jouer le rôle de médiateur entre les deux autres types d'agents mobiles.

Un agent acheteur véhicule avec lui le code d'un service offert par le fournisseur qu'il représente, ainsi que les données propre à ce service comme par exemple le prix du service (dans le cas ou le service n'est pas gratuit).

Un agent vendeur peut communiquer avec son fournisseur à distance afin d'accéder à des ressources supplémentaires.

Un agent dont le service devient obsolète sera détruit par le système.

Chaque service appartient à une classe de services. Un ensemble de classe de service constituent un domaine d'application.

Ainsi si on prend par exemple **la recherche d'image médicale** alors on peut la situé dans la classe de service **recherche des documents images** qui se trouve dans le domaine d'application **Recherche d'informations**.

Les services offerts doivent être identifiés, référencés et catalogué : c'est le **catalogue des services** ou **SVC** consulté par les utilisateurs du système.

4.1. Les MarketPlaces :

Une MarketPlace (MP) est un ensemble de machines interconnectées (LAN) doté :

- D'un pare-feu (Firewall) pour garantir la sécurité interne du réseau.
- D'une base de connaissances qui joue le rôle d'un annuaire interne à la MP.

Les différents composants de la MakretPlace sont:

- **Les agents Facilitateur (AFMP) :** un agent qui à le rôle principale la gestion de la MarketPlace et de mettre à jour l'annuaire.
- **Base de connaissance :** la basse de connaissance de la MP, elle contient des informations sur les boutiques et les différents agents qui y résident.
- **Boutique :** Lieux de rencontre et de négociation et d'échanges entre les agents.

A l'arrivé, Le **routage** des agents vers les **Boutiques** adéquates est assuré par les agents facilitateur des MP.

Une **MarketPlace** ne contient que les services d'une même classe de services. Les services sont regroupés dans des boutiques. Chacune contient les agents vendeurs d'un même service. C'est au niveau des boutiques que les interactions entre les agents se font.

Voici un aperçu de la place MP :

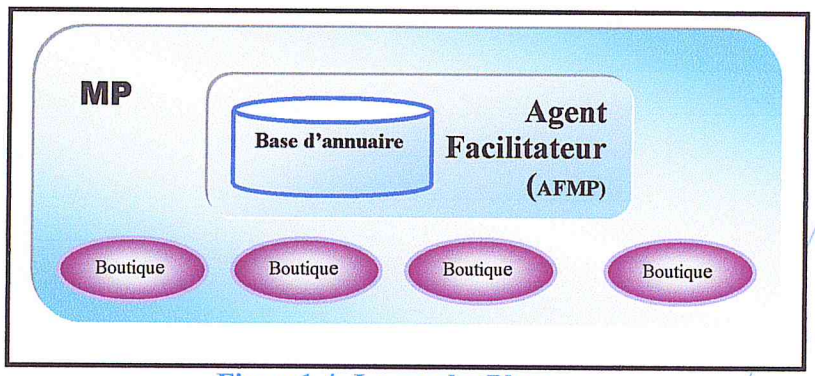


Figure1.4: La marketPlace

4.2. Localisation des MP :

Pour qu'un agent chercheur ou fournisseur d'un service puisse localiser les MarketPlaces hébergeant la classe de service à laquelle il appartient, il récupère les données de la base de connaissance qui dérive grâce au thème recherché. Comme les MPs sont identifiées par des adresses uniques (adresse IP) l'agent récupère une suite d'adresses IP des MarketPlaces appartenant à la classe de service en question. Cette suite constitue l'itinéraire de l'agent.

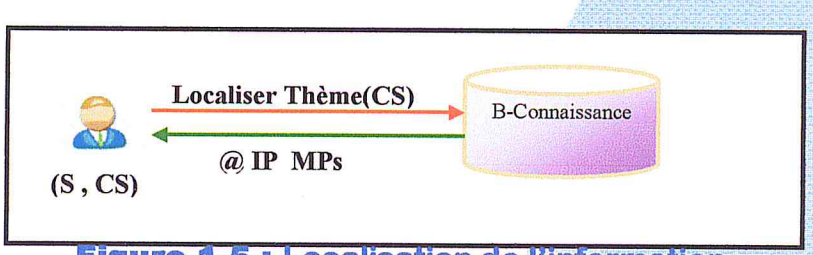


Figure 1.5 : Localisation de l'information

4.3. Agent Service Provider (ASP) :

Les ASPs sont des sites accessibles par internet qui permettent à l'utilisateur (clients ou services) de créer leurs agents qui les représentent dans le système.

Comme la création des agents par les utilisateurs eux même peut engendrer plusieurs problèmes au système notamment au niveau de la sécurité, alors il est conseillé de délégué la tâches de la création d'agents à un module de l'ASP. Dans ce cas la, un utilisateur demande à ce module de lui créer un agent avec les caractéristiques qui lui délivre (la requête plus précisément).

Un ASP contient les composants suivants :

- **ASP:** c'est un site responsable de la création des agents acheteurs ou vendeurs et les doter de clés publiques et privés:
- **SMJ (service de mise à jour)** : mentionne les services rendu à l'utilisateur en l'identifiant et le référenciant.
- Module de création des agents mobiles :

✓ **Les agents de recherche Méta-moteurs :**

Se sont des agents de recherche d'information, crées sur l'ASP leurs but c'est d'interroger plusieurs moteurs de recherche, filtrer les résultats et présenter les meilleur résultats a l'utilisateur

✓ **Les agents de recherche Mini-moteurs :**

Se sont des agents de fournisseur de service, crée sur l'ASP, porteurs de la base d'indexe qu'ils soient généralistes ou spécialisés (dans un thème donné d'information) récupérer a travers le formulaire remplie par l'utilisateur dotés d' :

1. Une base d'index : dont son contenu d'URL peut être transporté avec l'agent lors de sa migration.
2. Une procédure de recherche.

- **Base de connaissance de l'ASP :** Sur laquelle chaque utilisateur inscrit les informations sur l'agent qui lui a été créé.

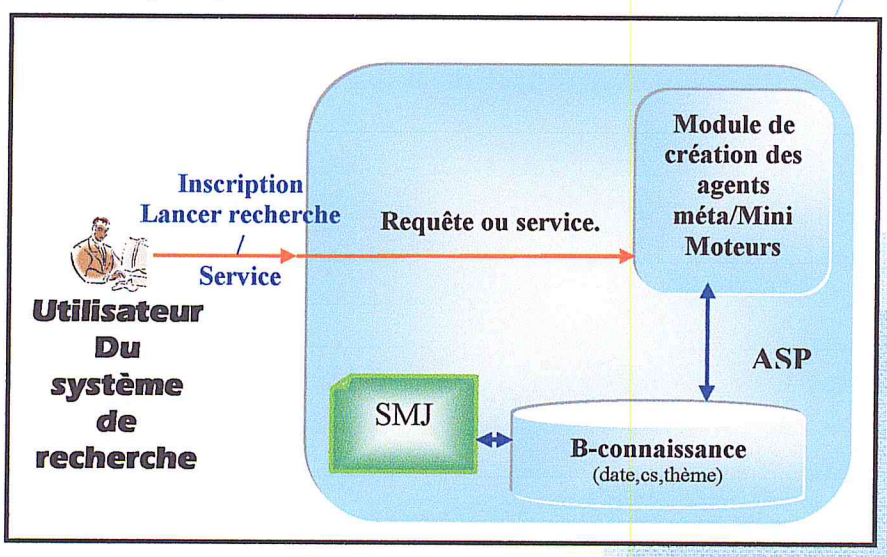


Figure 1.6 : ASP

5. La sécurité sur les MP :

L'infrastructure MP est doté d'une autorité de confiance de type **PKI** (Publique Key Infrastructure). Cette autorité est nommée **TSA** (Trust & Security Authority). Cette autorité a comme objectif de certifié tous les composants du système (ASP, MP, Agent, MPNS et la TSA elle-même).

Un agent doit être certifié avant de se déplacer dans le réseau, ce certificat délivré par la **TSA** est associé à la clé publique de l'agent que l'ASP lui délivre.

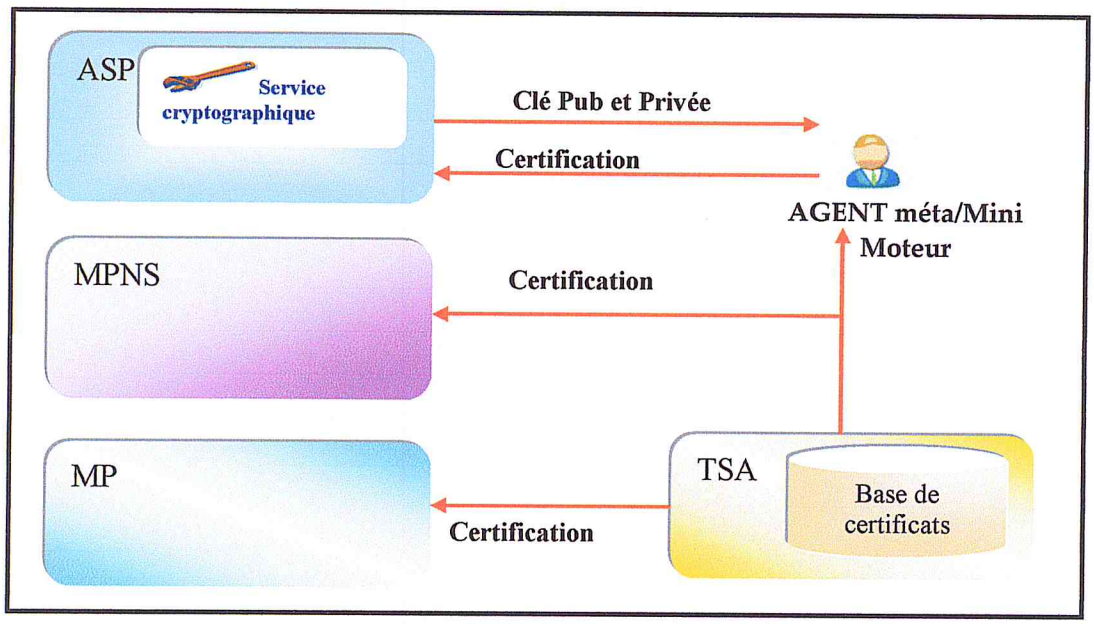


Figure 1.7: Fonctionnement de TSA

Un agent peut être signé et/ou crypté complètement ou juste partiellement.

5.1. La protection des sites ASP

Les sites ASP doivent être capables de prévenir les risques engendrés par le faite que le code d'un service fournit par un fournisseur soit malveillant. La prévention peut être faite par :

- Certification des fournisseurs par les autorités **TSA** ou une autorité **PKI** reconnue.
- Analyse du code du service offert avant de l'utiliser sur un agent vendeur.

5.2. La protection des MarketPlaces contre les agents

Les sites **MP** sont des sites hautement sécurisés grâce aux pare-feux qu'ils les protègent (**MPSS**). Ces pare-feux ont un accès aux certificats de la **TSA**.
A chaque fois qu'un agent arrive à une **MP** il sera rejeté s'il n'est pas certifié. Sinon s'il est certifié et son code est signé complètement ou partiellement, il (le code) sera analysé pour vérifier les signatures.

5.3. La protection des agents

La limitation de la mobilité des agents juste à l'ensemble des MarketPlaces délivré par la base de connaissance qui constitue l'itinéraire de l'agent et la certification des sites la base de connaissance réduit considérablement les risque, mais ces risque ne sont pas exclus de faite qu'un fournisseur d'un service peut fournir du code malveillant, ce qui peut infecter les MP d'où la possibilité d'infection des agents qui y migrent.

5.3.1. Utilisation d'un réseau fermé :

Le système se compose d'un réseau fermé, aucune entité extérieure n'est autorisée d'y accéder. Une entité qui veut utiliser le système (proposition ou recherche d'un service) doit déléguer un agent pour lui représenter dans le système.

5.3.2. Prévention de la falsification des agents :

Chaque agent dans le système (comme toutes autres entités dans le système) possède une paire de clé, une clé privée et une clé publique que le TSA lui délivre lors de sa création. Ces clés sont utilisées dans plusieurs opérations faites par l'agent. Dans le cas de la prévention de la falsification, un agent signe sa requête par sa clé privée. A son arrivée à un hôte, ce dernier contrôle la requête de l'agent on récupérant sa clé publique auprès de la TSA.

En plus, après avoir trouvé des résultats, ils seront tous d'abord signés par l'entité qui les délivre, ensuite ils seront cryptés par la clé publique de l'agent.

6. Architecture IMAN (Interaction Mobile Agent in Nail):

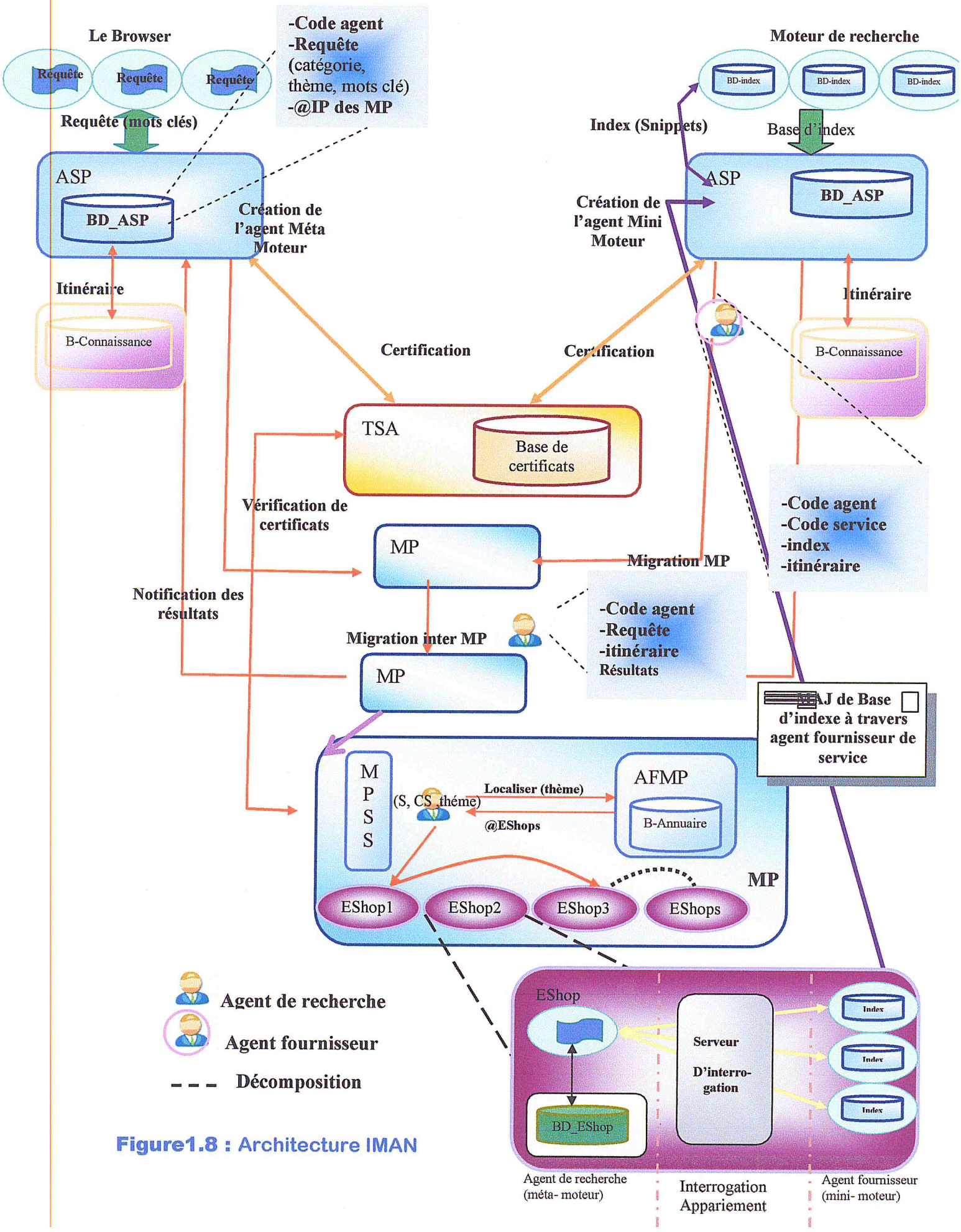


Figure1.8 : Architecture IMAN

Les systèmes de la recherche d'information sur internet doivent contenir trois éléments essentiels suivants :

- Les clients qui font des requêtes de recherche.
- Un moteur de recherche (ou annuaire) qui comprend une base d'index sur les documents.
- Un spider qui scrute continuellement le web afin de garder les bases d'index à jour.

Les méthodes utilisées par les spiders sont déférentes d'un moteur de recherche à un autre. Dans notre système, un agent de recherche (agent acheteur) interroge plusieurs moteurs de recherche, filtre les différents résultats et restitue un meilleur résultat à l'utilisateur. Cet agent est dit **méta-moteur de recherche**.

Tandis qu'un agent vendeur (fournisseur d'un service) représente un mini-moteur de recherche car il est muni d'une procédure de recherche qui se limite à l'index qu'il véhicule.

Le **SVC** (catalogue des services) utilisé dans le système est de la forme suivante :

Domaine=RI **Classe de service**= Catégorie de recherche
Service (final) =Thème de recherche

Exemple

Domaine= RI **Classe de service**= Général **Service**= Général
Domaine= RI **Classe de service**= IT **Service**= Software

Ce qui fait que les agents acheteurs (méta-moteurs) et les agents vendeurs (mini-moteur) se rencontrent dans des MarketPlace classées selon les classes de service.

L'agent acheteur ne porte avec lui que la requête du client. Cette requête est traitée au niveau de l'ASP afin de déterminer les classes de services adéquates. Si la requête ne peut pas être classée dans aucun des classes existantes, la classe généraliste sera la classe attribuée par défaut.Chaque agent se voit attribuer un itinéraire qui contient les MarketPlace de sa classe de service.

- La fonction de la recherche dans les index est différente d'un agent à un autre, il se peut que deux agents aient le même index, mais les deux délivrent des résultats différents.

Dans ce cas la le système multi-agents contient des agents chercheur (méta-moteur) légers (qui ne contient pas une fonction de recherche) et des agents vendeurs menés d'une fonction de recherche (mini-moteur).

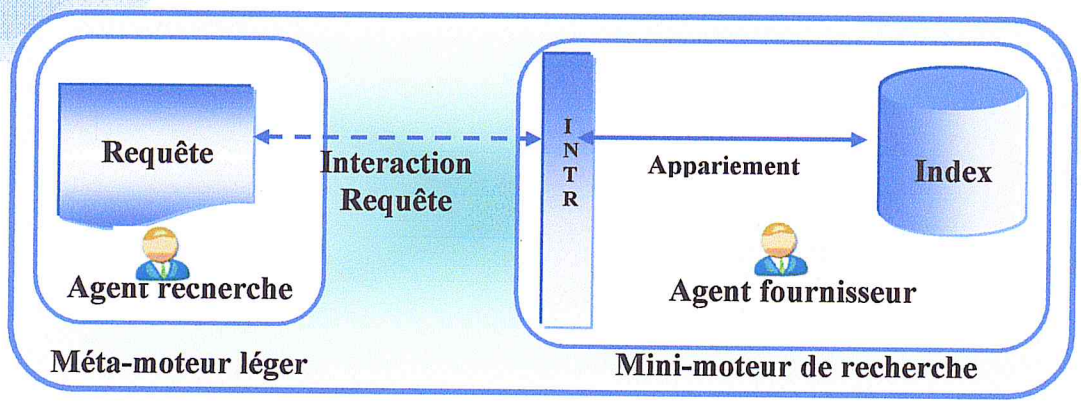


Figure1.9 .1: Recherche au niveau de l'agent fournisseur de service

- Le déplacement de la fonction de recherche de l'agent fournisseur du service à l'agent chercheur (qui devient lourd), permet d'augmenter la qualité des résultats, de faite qu'une tel pratique va augmenter la concurrence entre les agents fournisseurs de services pour offrir le meilleur index sans se préoccuper de la qualité de la fonction de recherche.

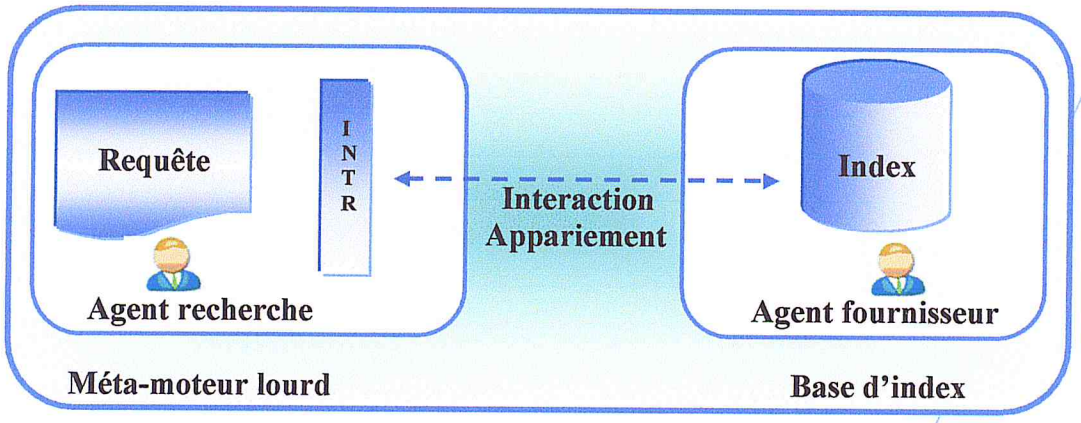


Figure1.9.2 : Recherche au niveau de l'agent de recherche d'information

Cette solution va apporter des gains considérables au système en générale, mais on peut toujours l'améliorer, on mettant la fonction INTR (interrogation) au niveau de l'e-shop et non pas au niveau de l'agent chercheur. Ceci apporte les avantages suivants :

- une plus grande légèreté pour l'agent chercheur, donc diminution du trafic sur le réseau et diminution de la bande passante consommée.
- Possibilité d'intégrer une variété de fonctions interrogation selon les différents types de recherche. Par exemple une fonction pour la recherche de documents textuels, une deuxième pour la recherche des documents image, une troisième pour les documents vidéo et une quatrième pour les documents audio et ainsi de suite.

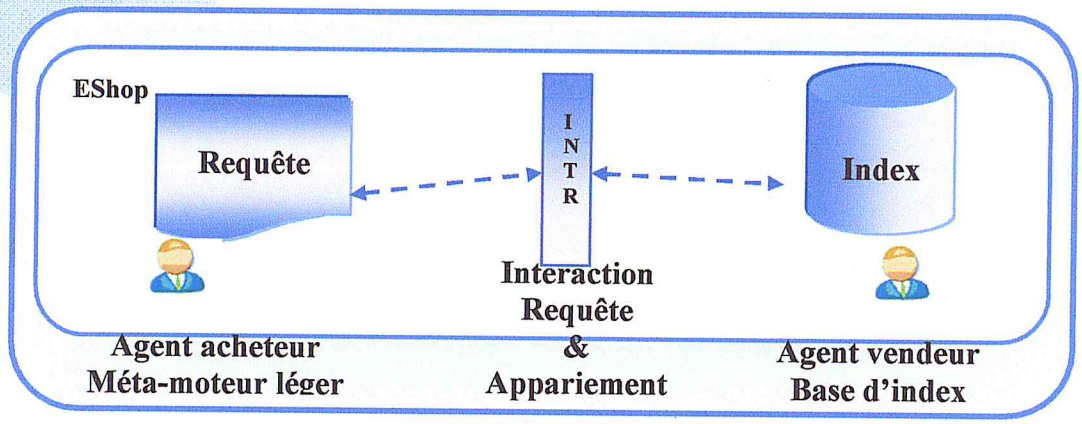


Figure1.9.3 : Recherche sur la boutique

6.1. Interaction sur la Marcket Place :

De ce qui est cité précédemment nous pouvons conclure un processus de fonctionnement sur la place marchée comme suit

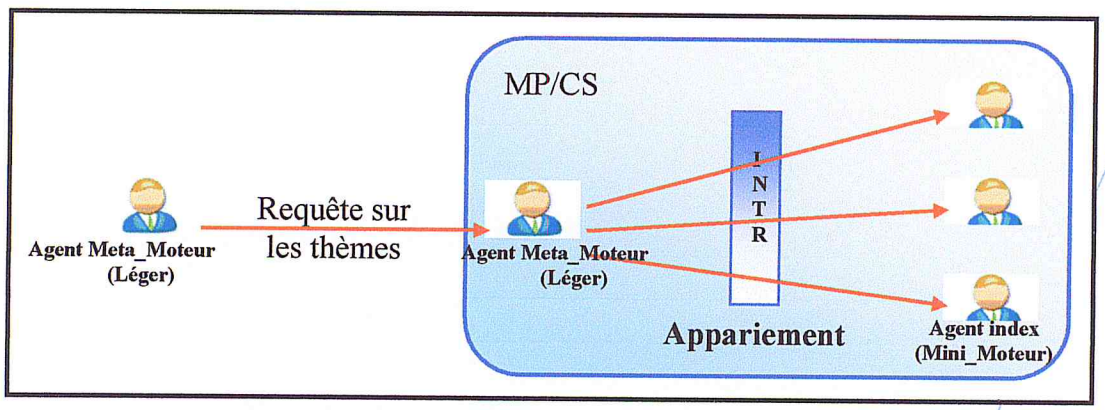


Figure1.10 : Interaction sur la Marcket Place

6.2. Mise à jour des index :

C'est le fait que l'utilisateur (fournisseur) met périodiquement sa base d'indexe en envoyant des mises à jours si nécessaire en passent par son ASP afin de décider de la validité de son information

7. calcul de taux de satisfaction d'un agent de recherche

Il semble que la réponse à cette question n'est pas aussi facile que de la poser. Le faite de rendre l'agent capable de décider lequel des résultats est bon et lequel ne l'ai pas nécessite le développement d'une intelligence artificielle très avancée, en plus personne ne la encore fait parce que ce n'est pas possible, au moins actuellement, de remplacer l'analyse humaine par une analyse automatisé.

Pour cela on propose une solution qui peut aider à résoudre au moins partiellement ce problème. On a tenté d'hybrider trois méthodes qui donne chacune seule s'assez bons résultats :

- Le nombre de clients pour un fournisseur.
- La réputation des agents fournisseurs calculée automatiquement.
- La réputation des agents fournisseurs calculée selon le comportement de l'utilisateur.

7.1. Le nombre de clients pour un fournisseur

Le nombre d'agent chercheurs négociant simultanément avec un fournisseur de service peut donner une bonne idée sur la qualité du service fournit. Plus le nombre augmente plus la probabilité que la qualité du service augmente. D'ailleurs c'est ce qui utilisé par les gens qu'on ils vont pour s'acheter un produit dans un marché contenant plusieurs magasins qui fournissent le même produit, la plupart se dirige intentionnellement vers les magasins ayant le plus grand nombre de clients.

Pour garantir de meilleurs résultats, on va limiter le nombre maximum d'agents client chez un fournisseur et ceci dans le but de :

- Donner une chance aux fournisseurs qui ont un nombre inferieure de clients.
- Garantir un certain niveau de condition du service (par exemple un temps de service raisonnable).

7.2. La réputation des agents fournisseurs calculée automatiquement

Le système dispose d'un endroit ou les agents chercheurs peuvent noter certains critères chez les agents fournisseurs comme par exemple la disponibilité du service qu'il cherchait ou le temps de réponse à une demande. Cela peut aider d'autres agents à décider d'aller négocier avec tel ou tel agent.

7.3. La réputation des agents fournisseurs calculée selon le comportement de l'utilisateur

Il serait meilleur d'utiliser les feedbacks des utilisateurs pour évaluer les agents, mais l'expérience à montrer que l'utilisateur ne s'intéresse pas vraiment à retourner ce genre de feedback dans le domaine de la recherche d'information. Pour cela en analysant le comportement de l'utilisateur on peut avoir une idée sur la qualité des documents qu'il trouve. On peut prendre comme critère d'analyse :

1. L'ouverture du document après avoir lu la description.
2. Le temps passé en lecture du document.
3. La vitesse de défilement du doucement.
4. Le copiage d'une partie de document.

8. Schéma d'un scénario d'un agent Fournisseur d'index :

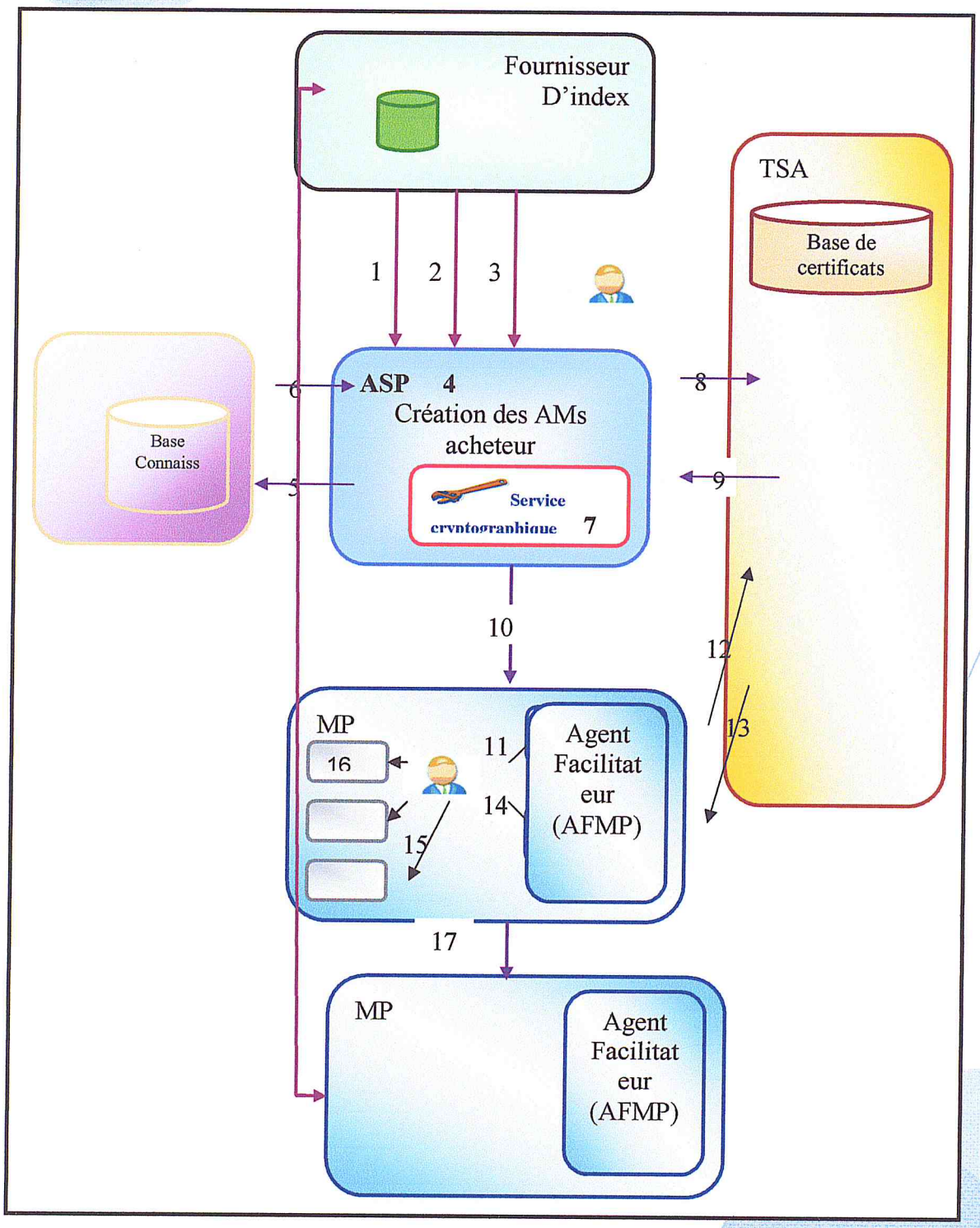


Figure 1.11.1 : Schéma d'un scénario d'un agent Fournisseur d'index

L'architecture se déroule selon le scénario suivent :

1. connexion et authentification au niveau de l'ASP
2. proposition du service
3. déconnexion du fournisseur
4. création d'un agent fournisseur
5. Demande l'itinéraire des MPs a la base de connaissance
6. Transmission de cet itinéraire
7. génération de clé
8. demande de certificat de clé de l'agent fournisseur au TSA
9. accord de certificat
10. migration de l'agent
11. filtrage de l'agent
12. Demande de la clé publique
13. envoie de la clé
14. l'agent fournisseur communique avec les MPSM
15. Agent fournisseur déplace vers les EShops et reste
16. l'Agent fournisseur reste sur les EShops tout son Time To Live (TTL)
17. communication avec le fournisseur pour mettre a jour la base d'index

9. Schéma d'un scénario d'un agent de Recherche d'information :

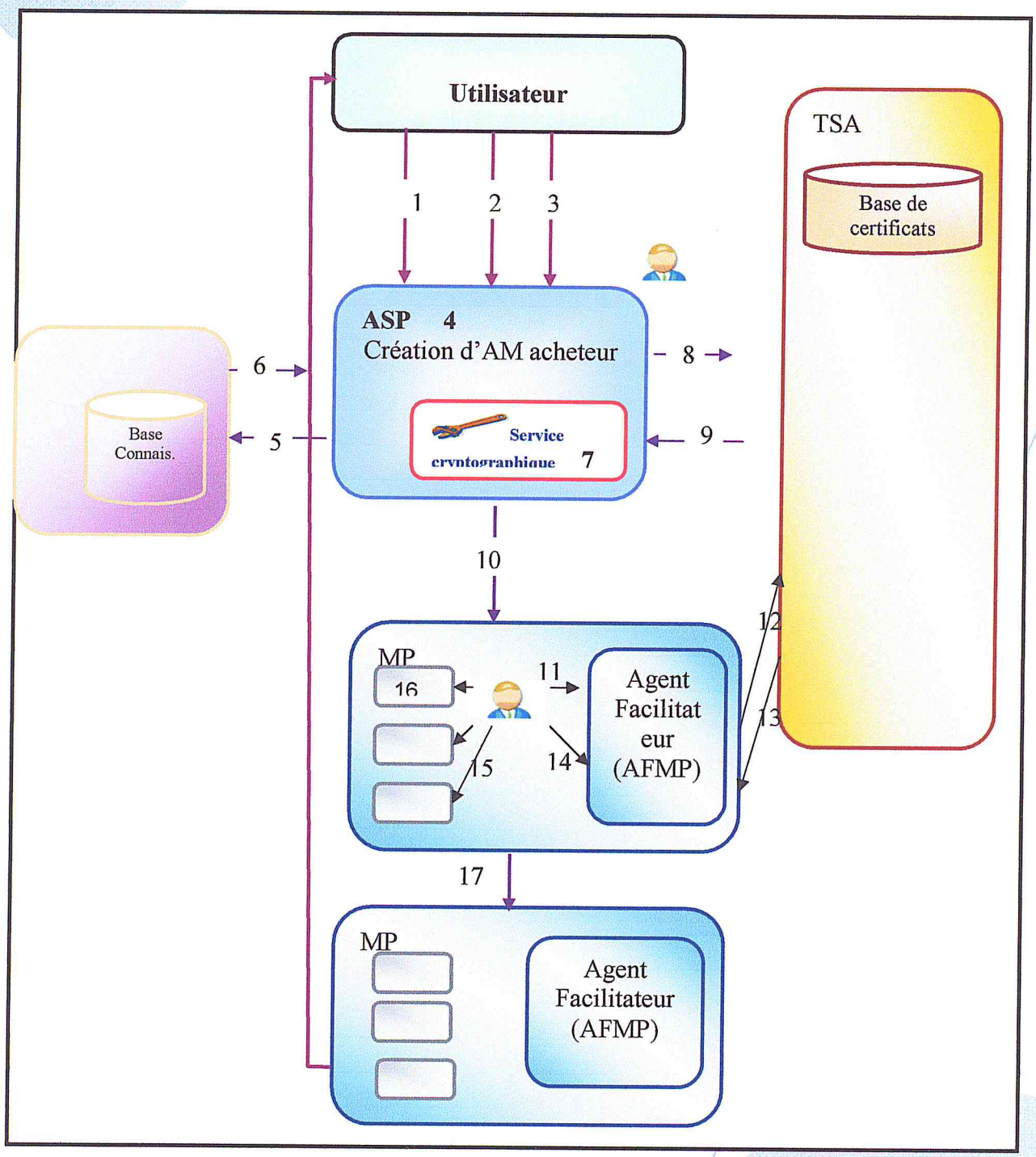


Figure 1.11.2 : Schéma d'un scénario d'un agent Recherche d'information

L'architecture se déroule selon le scénario suivent :

1. connexion et authentification au niveau de l'ASP
2. Envoi de la requête
3. déconnexion du client
4. création d'un agent de recherche véhiculant la requête
5. Demande l'itinéraire des MPs a la base de connaissance
6. Transmission de cet itinéraire
7. génération de clé
8. demande de certificat de clé de l'agent de recherche au TSA
9. accord de certificat
10. migration de l'agent
11. filtrage de l'agent
12. Demande de la clé publique
13. envoie de la clé
14. l'agent de recherche communique avec l'agent-Facilitateur
15. Agent de recherche déplace vers les boutiques
16. l'Agent de recherche déplace entre les MPs.
17. retour vers le client

II. Conception du system :

1. Cas d'utilisation

Dans cette partie nous allons définir les fonctionnalités vues directement par les utilisateurs du système, les clients et les fournisseurs de services. La figure suivante représente le diagramme des cas d'utilisations.

1.1. Identification des cas d'utilisation :

Dans ce qui suit nous allons définir les fonctionnalités vues par les utilisateurs du système et les fournisseurs de services.

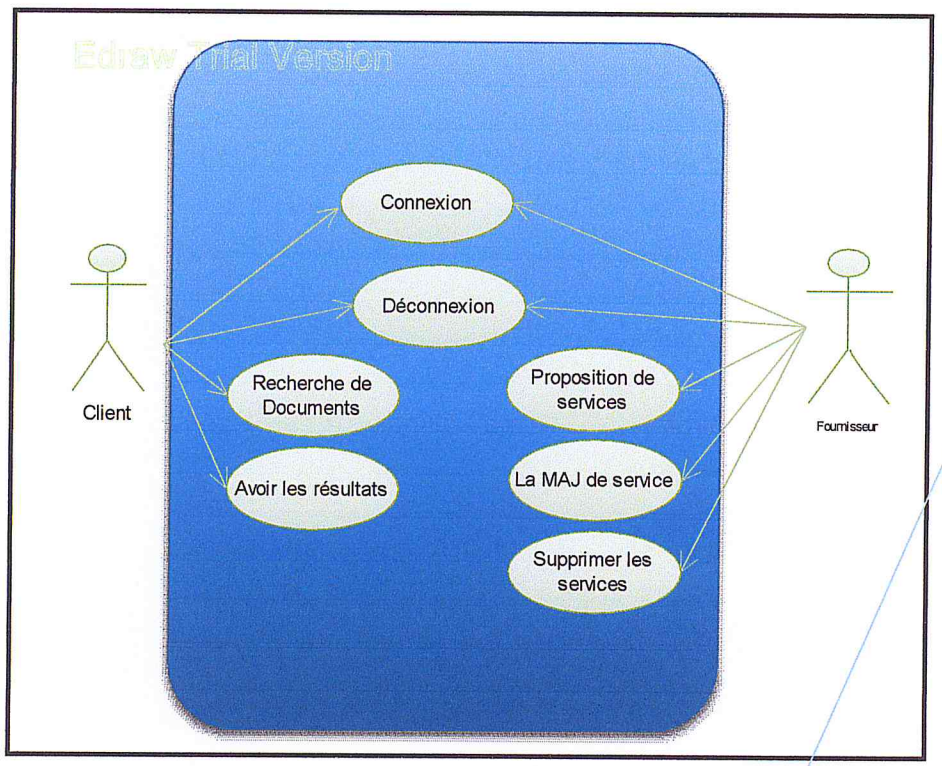


Figure 2.1 : Diagramme Use Case

1.2. Scénario de cas d'utilisation :

Le client : 1) Se connecte à notre système :

Ouvrir une fenêtre principale

Remplie le formulaire d'utilisateur: inscription

Lancer la validation

2) Se déconnecte de notre système :

Le client peut se déconnecte du système

3) Recherche des documents :

Le système affiche la recherche

Le client sélection le résultat adéquat a sa recherche

Le client relance une nouvelle recherche s'il désire

4) Voir les résultats :

Le client reçoit ces résultats

Le client récupère ces documents

Le fournisseur : 1) Se connecte à a notre système :

Ouvrir une fenêtre principale

Remplie le formulaire d'utilisateur: inscription

Lancer la validation

2) Se déconnecter du système :

Le fournisseur peut se déconnecte du système

3) Proposition de service :

Le fournisseur demande à ajouter un service

Le système affiche le formulaire

Le fournisseur le remplie les coordonner du service et valide

Le système ajoute le nouveau service

4) Mettre à jour le service :

Le fournisseur demande à mettre à jour le service

Le système affiche le formulaire

Le fournisseur le remplie les mises à jour du service et valide

Le système valide les mises à jour

5) Supprimer le service :

Le fournisseur demande à supprimer un service

Le fournisseur le supprime le service voulu et valide

2. Description des agents système :

Le système compte deux types d'agents :

2.1. Agents mobiles (Agent_User) :

Se sont les agents de recherche (AR) et les agents fournisseur (AF) qui représentent respectivement les clients et les fournisseurs de services.

2.2. Agents facilitateurs (Agent_F) :

Se sont les différents agents qui résident au niveau des MPs pour faciliter le travail des agents mobiles et assurer le bon fonctionnement du système.

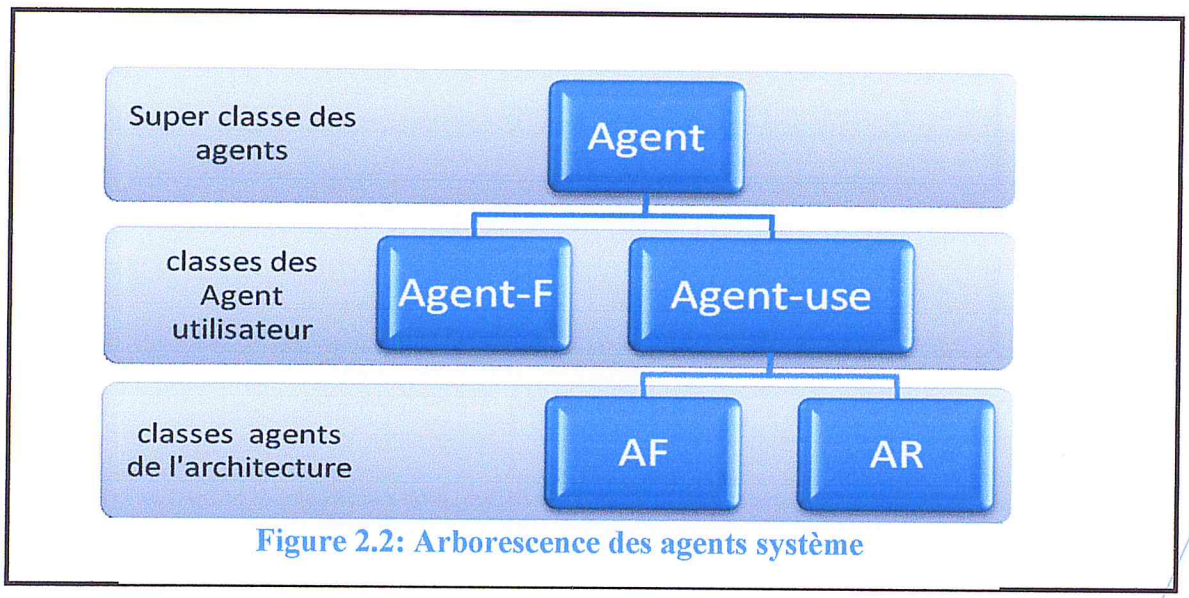


Figure 2.2: Arborescence des agents système

3. Les interactions entre les composants système:

Nous allons voir maintenant comment ces communications se font. Pour cella, on a besoin d'utiliser le diagramme de séquence de l'AUML.

Comme la seule différance entre le comportement des A Recherche et les A Fournisseur de service se situe dans la négociation au niveau des boutiques alors on va utiliser le nom commun Agent-User pour représenter les deux dans les cas au leurs comportements sont identiques.

3.1. Niveau 1 : Interaction sur l'ASP :

Lors de la création de l'agent au niveau de l'ASP, il doit récupérer l'itinéraire qui correspond à sa requête avant de migrer vers les MPs. Pour cela, l'agent doit faire une demande à une B-Connaissance contenant ces itinéraires.

Une fois que l'agent récupère l'itinéraire, il demande cette fois-ci au TSA pour lui fournir un certificat, ensuite il la passera à l'agent.

A ce stade la, l'agent peux entamer sa tournée dans le système. Il entre toute d'abord dans une négociation avec les Agent-F de la première MP qui figure dans son itinéraire. Si la demande de migration est acceptée, l'agents-F lui fait une demande d'authentification, et sure la réponse de cette demande l'agents-F va décider s'il acceptera la migration ou pas.

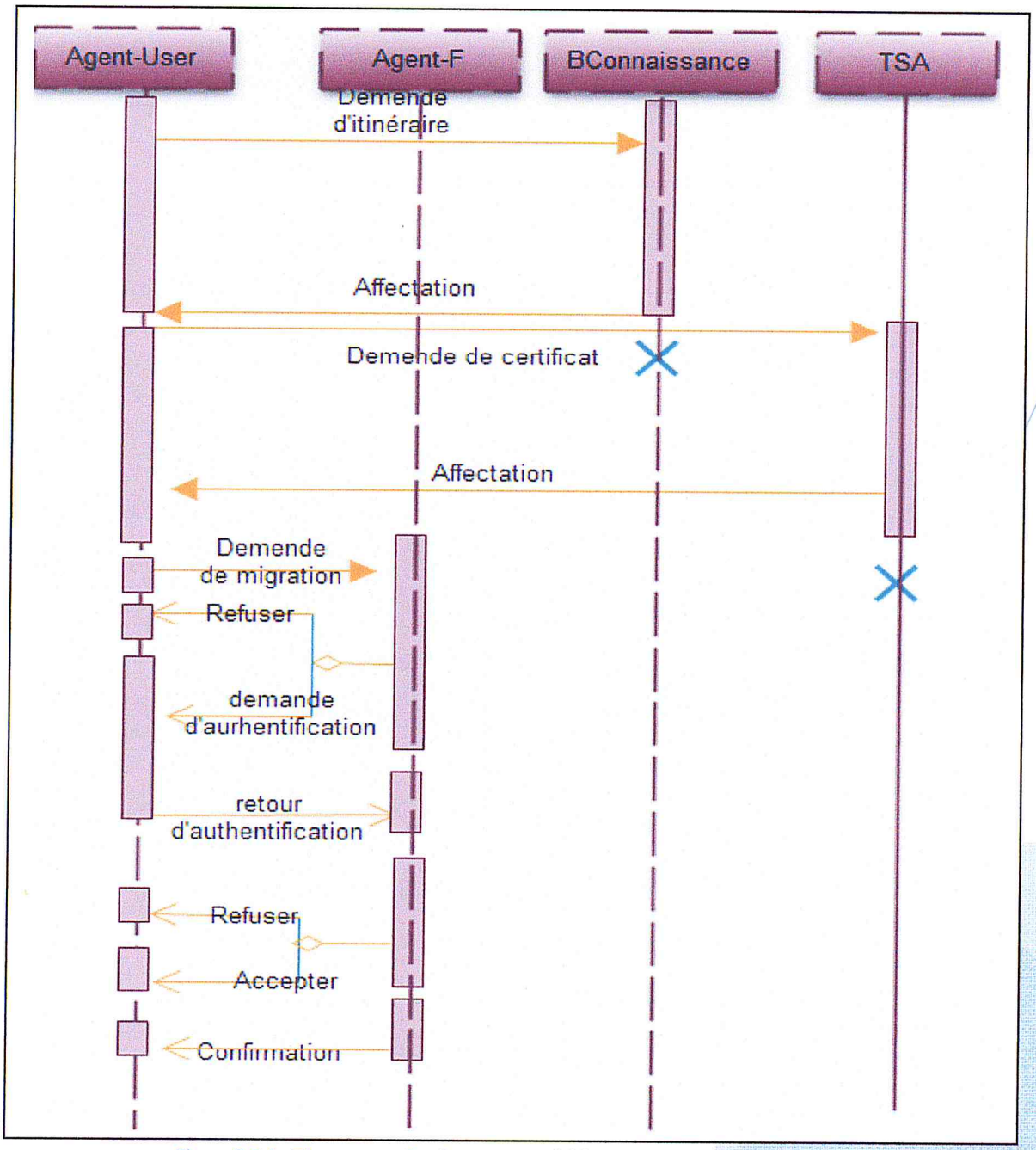


Figure2.3.1 : Diagramme de séquence sur l'ASP entre agent User et les différents Agents des sites

Lorsqu'un utilisateur désire lancer une recherche ou un service, l'agent utilisateur envoi une demande vers la base de connaissance qui se trouve sur l'ASP afin de migrer vers les MPs qui va lui attribuer un itinéraire, ensuit il demande à être certifier par la TSA, qui va lui accorder une clé de certification

L'agent demande a l'agent facilitateur de migrer qui va lui envoyer un message de demande de certificat, ce certificat peu être refuser ou accepter si oui l'agent facilitateur lui envoi une confirmation.

3.2. Niveau 2 : interaction sur les MPs :

L'agent-user demande a l'agent-F de lui filtrer cette demande serra accepter ou refuser selon la recherche si oui, l'agent-User il envoie le thème de recherche vers Base de connaissance qui lui répondra par les noms des boutiques afin de migrer.

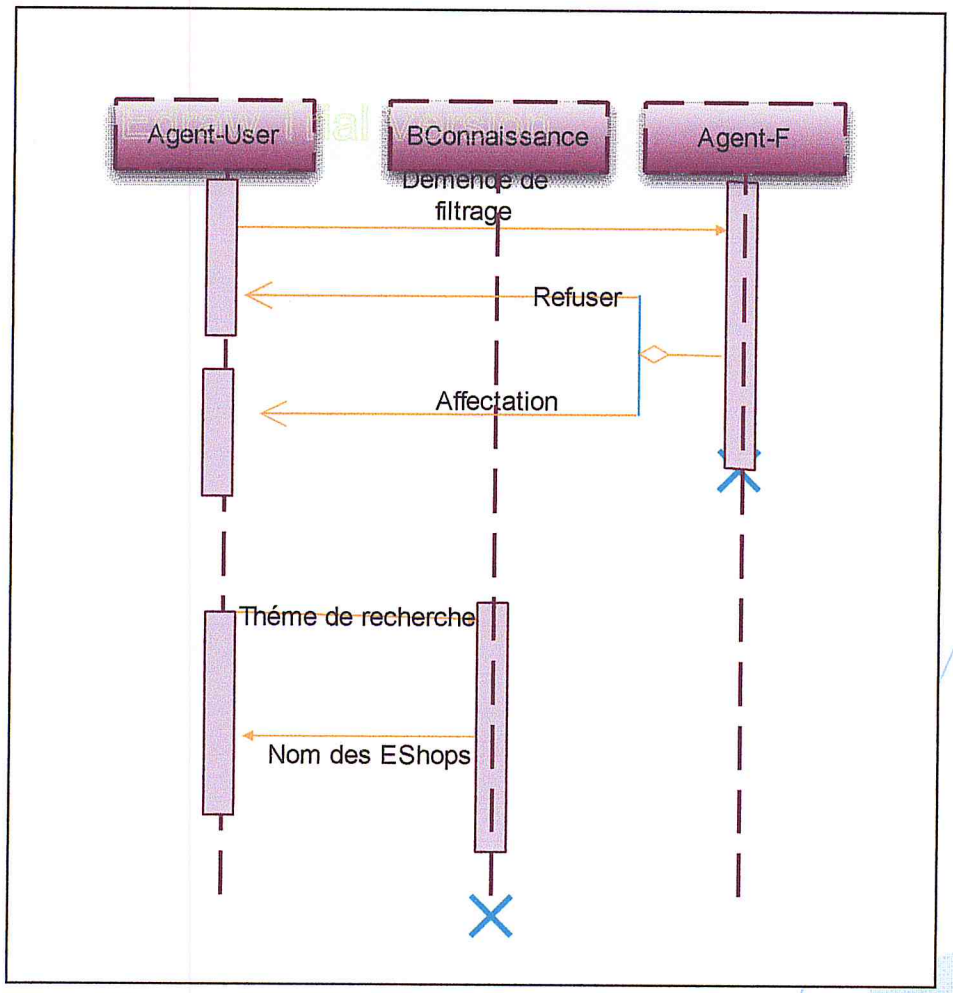


Figure2.3.2 : Diagramme de séquence sur la MP

3.3. Interaction entre les agents de recherche et les agents fournisseurs de services au niveau des e-shop :

Dans les systèmes multi-agents, la négociation est une composante de base vue l'autonomie des agents. Cette négociation se base essentiellement sur la communication et la prise de décision. Il existe plusieurs types de négociation entre les agents citant par exemple :

3.3.1. Les enchères :

Les enchères sont des mécanismes d'interaction assez simple. Une enchère contient un **initiateur** (actionner) qui annonce un objet à vendre et que veut le vendre au prix le plus élevé, et des **participants** (Bidders) qui sont intéressés à l'acheter et qui veulent l'acheter au plus petit prix possible. Chaque agent travail pour soie, ce qui élimine toute coopération entre eux.

3.3.2. L'allocation des tâches par réseau contractuel (contract-net) :

Il existe plusieurs protocoles qui ont été conçus principalement pour en vue de la coordination des agents coopératifs ayant des buts en commun. Parmi les tous premiers protocoles de ce type on trouve **Contract-net** (protocole de réseau contractuel) qui contienr deux type d'gents: **gestionnaire** et **contractant**. L'agent le gestionnaire commence par décomposer cette tâche en plusieurs sous-tâches. Le gestionnaire annonce chaque sous-tâche sur un réseau d'agents contractants. Les agents qui reçoivent une annonce de tâches à accomplir évaluent l'annonce.

Les agents qui ont les ressources appropriées, l'expertise ou l'information requise pour accomplir la tâche, envoient au gestionnaire des soumissions (bids) qui indiquent leurs capacités à réaliser la tâche. Le gestionnaire rassemble toutes les propositions qu'il a reçues et alloue la tâche à l'agent qui a fait la meilleure proposition. Ensuite, le gestionnaire et les contractants échangent les informations nécessaires durant l'accomplissement des tâches. Par exemple, le contractant annoncera au gestionnaire quand l'exécution de la tâche sera terminée.

Voici un schéma qui résume la négociation par FIPA-contract-net

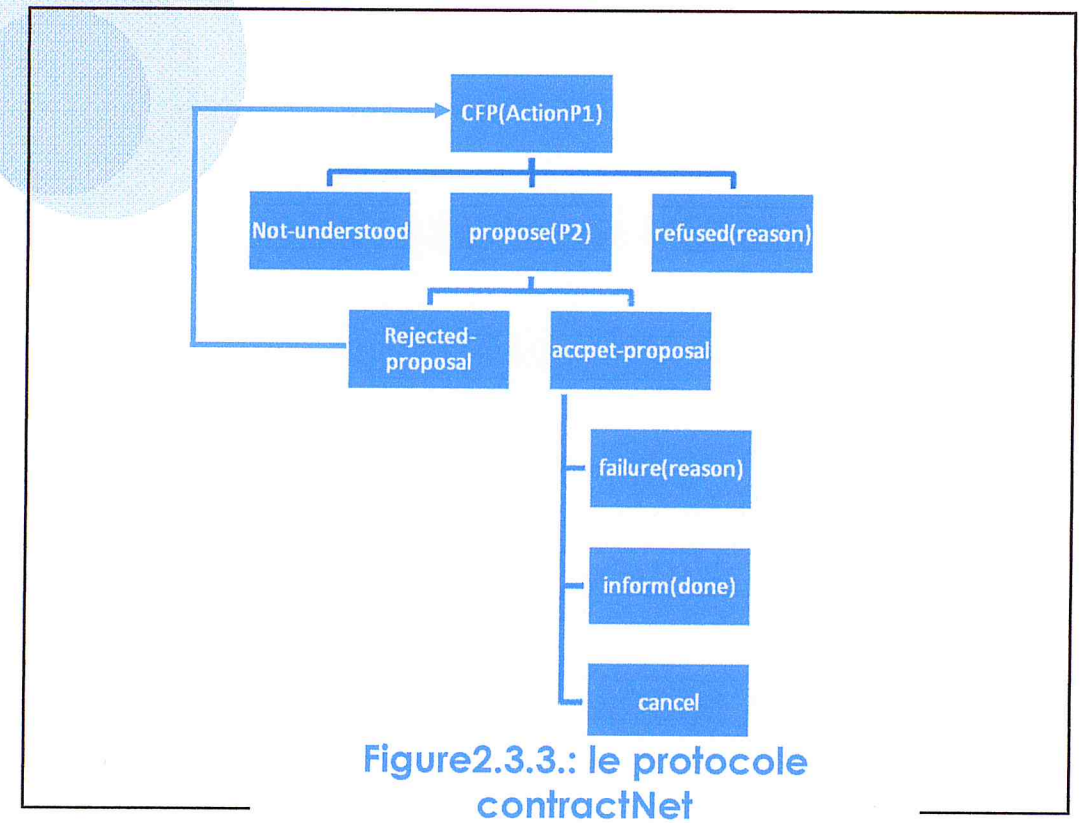


Figure2.3.3.: le protocole contractNet

On voit bien que ce type de protocole convient bien à la négociation entre les agents pour faire de la recherche d'informations en générale et à la négociation entre les agents chercheurs et les fournisseurs de services dans notre cas. Pour cela on va l'adapter pour réaliser et représenter les négociations entre les agents dans le système.
Quant a notre systeme un Agent mobile vendeur arrive dans une boutique, il demande au boutique manger de le mettre en contacte avec les agent mobile acheteur qui correspond à sa requête. La négociation entre ces deux types d'agents se fait de la manière suivante :

L'agent chercheur envoie à tous les agents fournisseurs de services le nom du service dans lequel la requête se situe, ceux qui on dispose le signalent à l'agent chercheur qui envoie à son tour les mots clés constituant la requête aux agents qui ont répondu positivement.

A ce niveau de la négociation, les agents fournisseurs de services envoie à l'agent chercheur les conditions du leurs services.

Par les conditions du service en comprend la qualité, le prix (si le service est payant) et d'autres informations relatives au service. L'agent chercheur a le choix de refuser les conditions d'un agent fournisseur d'un service donc ne rein récupérer de la négociation entre eux, ou de les acceptées et en contre partie l'agent fournisseur de service lui délivre les index correspondants à sa requête.

3.4. Niveau 3: interaction sur les boutiques :

L'agent de recherche d'information envoi le nom de service désiré a l'agent fournisseur. L'agent fournisseur lui envoi une réponse négative ou une réponse positive si oui l'agent de recherche envoi une requête de mots clés puis lui transmet les conditions de service exemple le prix de service si le service est payant, l'agent de recherche refuse ou accepte si les conditions sont acceptées il lui renvoi les index

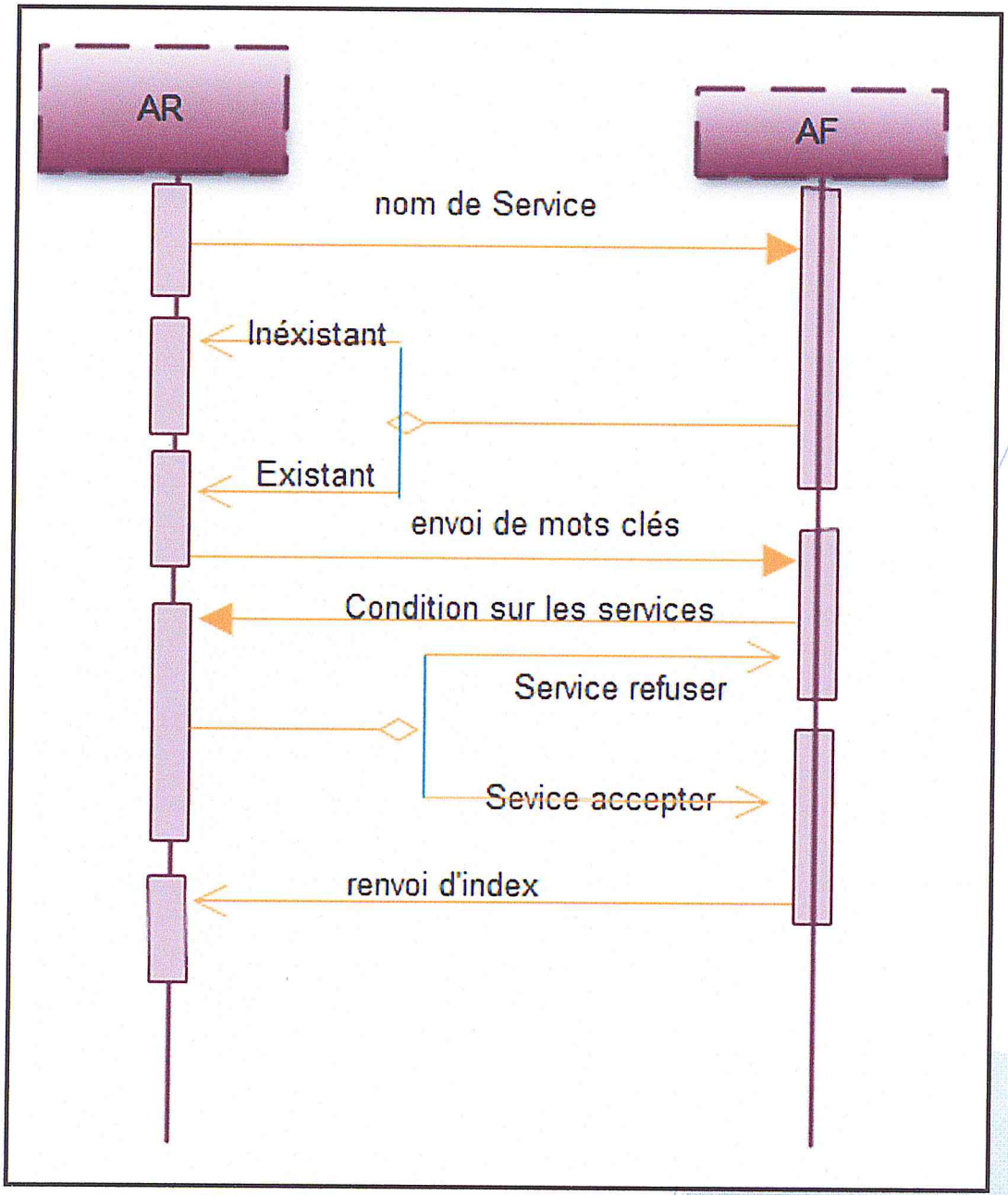


Figure2.3.4: Diagramme de séquence entre AR et AF

4. Diagrammes de classes des agents du système

Cette partie va faire l'objet des diagrammes de classe des différents agents de système. Se diagramme qui va montrer l'échange de message entrant et sortant sur les agents de système.

4.1. Agent facilitateur Agent-F :

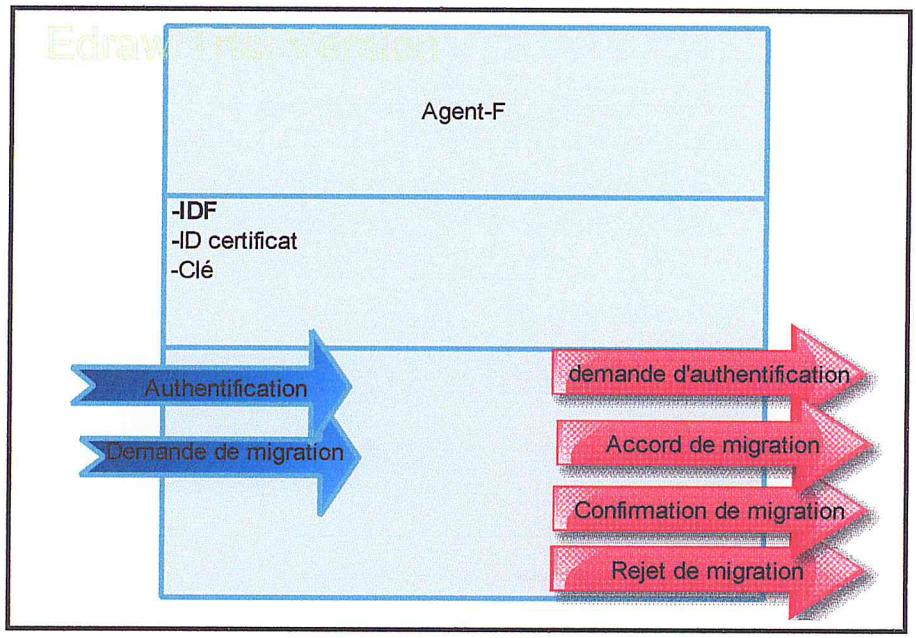


Figure2.4.1 : diagramme de classe AUML de l'agent Facilitateur

IDF : L'identifiant du l'agent du type facilitateur

ID_certificat :L'identifiant du certificat de l'agent-F

Clé : La clé publique utilisée par l'agent-F.

4.2. Agent utilisateur Agent-user :

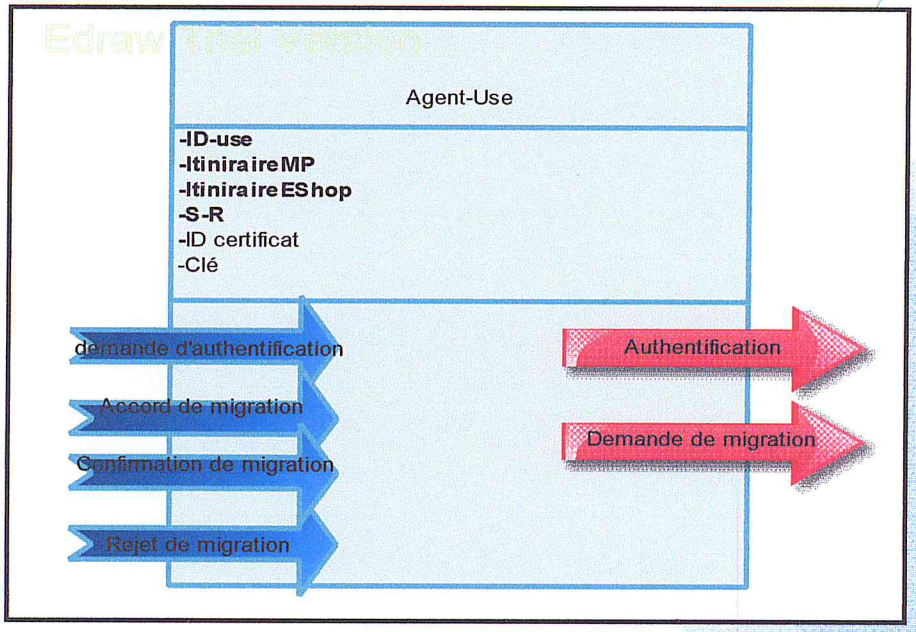


Figure2.4.2 : diagramme de classe AUML de l'agent utilisateur

ID_use : L'Identifiant du l'agent.

ItinéraireMP : itinéraire de l'ensemble des MP à visiter. Cet itinéraire est délivré par le MPNS.

ItinéraireBoutique : L'itinéraire de l'ensemble des boutiques à visiter. Cet itinéraire est délivré par le MPDS du Merket Place MP.

S-R : Il représente le code du service dans le cas d'un agent vendeur, et du code de la requête dans le cas de l'agent acheteur.

IDcertificat : L'identifiant du certificat de l'agent-Use.

Clé: La clé utilisé par l'agent-Use que le TSA lui délivre.

4.3. Agent Fournisseur AF:

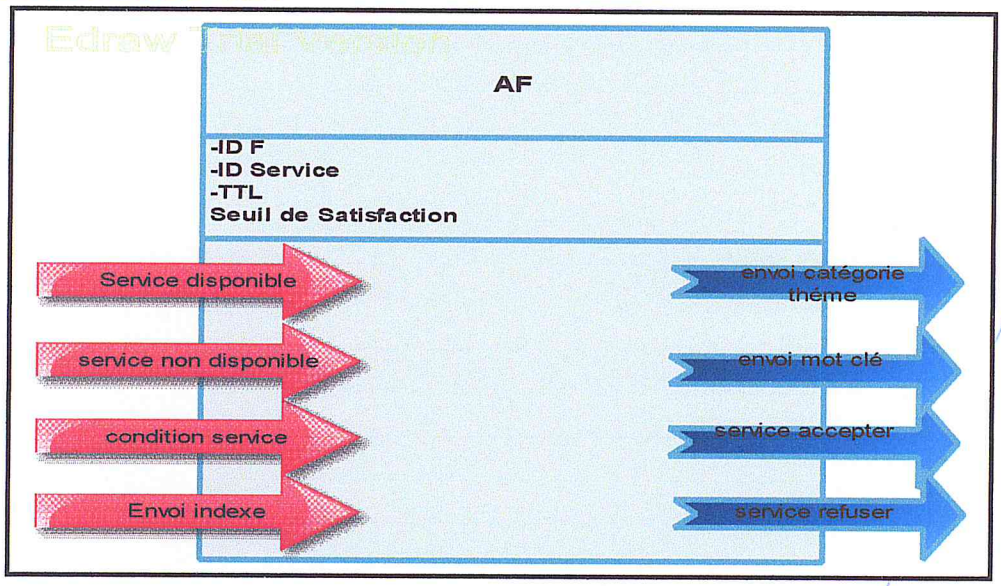


Figure2.4.4 : diagramme de classe AUML d'agent Fournisseur

ID F: L'identifiant de l'agent.

ID service : L'identifiant du service de l'agent

TTL : Time To Live : c'est le temps Durant lequel l'agent vie dans le système et doit quitter avant sa fin

Seuil de satisfaction : Si la mesure de la satisfaction de l'agent envers les résultats trouvé par rapport a sa requête.

4.4. Agent Recherche d'information AR:

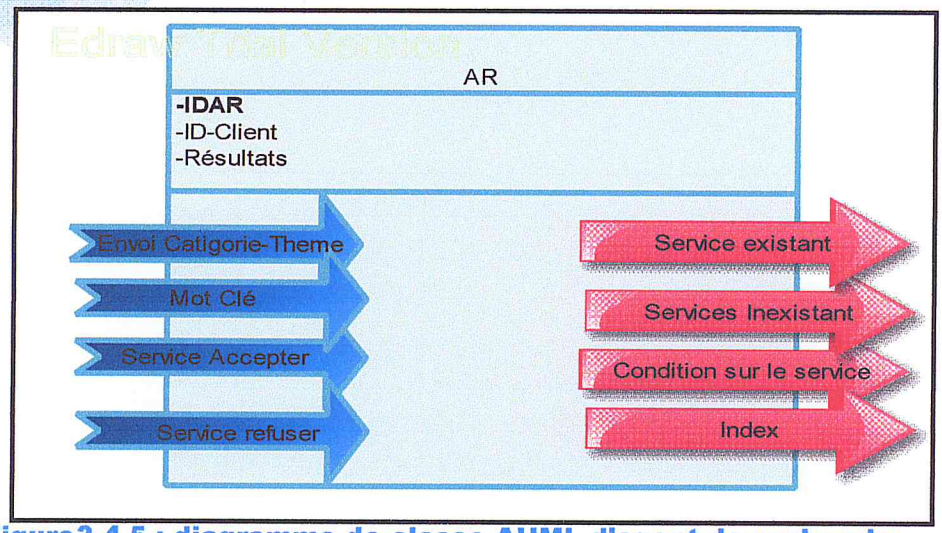


Figure2.4.5 : diagramme de classe AUML d'agent de recherche

IDAR : L'identifiant de l'agent AR.

ID_client : L'identifiant du client que l'agent représente

Résultats : Le résultat de la recherche de l'agent

5. Les bases de connaissances du système :

Pour des raisons de traçabilité du système entre autres les agents mobiles, les classe de service hébergés dans chaque Market place, on a construit une base de connaissance qui représente ces données. Ces bases de connaissance sont :

5.1. DB_ASP (la base de données de l'ASP) :

C'est la base qui se charge de garder les informations importantes au niveau de l'ASP. On peut l'appeler une base de données superviseur. Cette étape permet d'identifier les classes pertinentes de la DB_ASP qui sont :

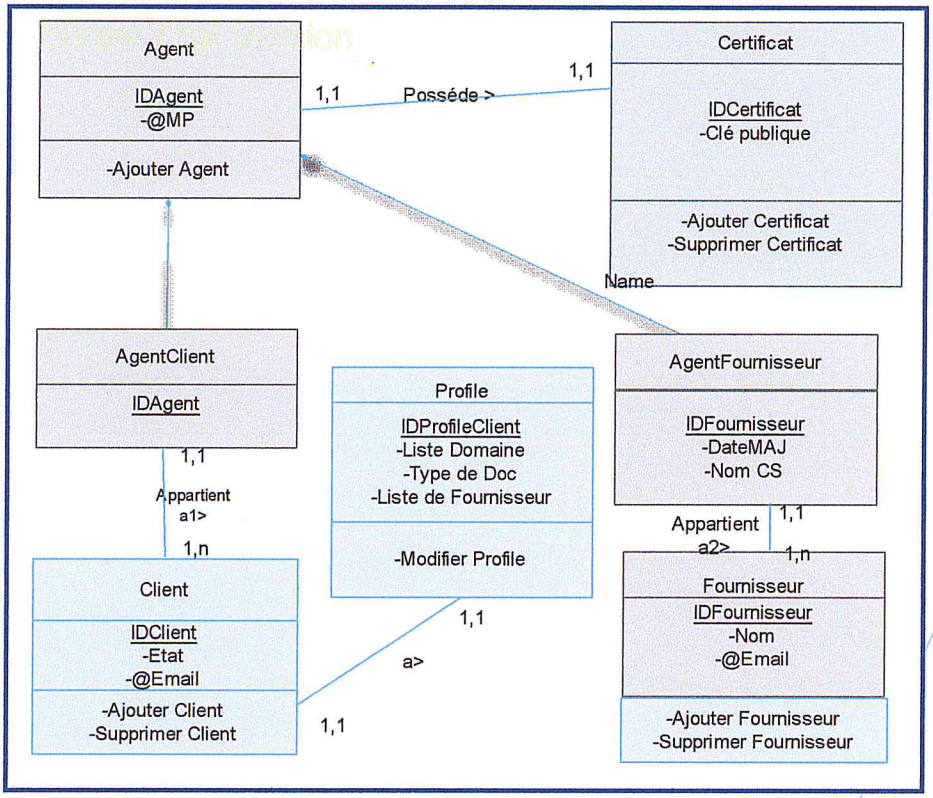


Figure2.5.4 : diagramme de classe sur la BD-ASP

Classe agent : Cette classe contient les informations sur tous les agents (chercheurs, fournisseurs et de services) qui vivent dans le système.

Classe certificat : Elle contient les informations sur la certification de tous les agents du système.

Classe agentClient : Elle contient les informations sur les agents chercheurs du système. Elle hérite de la classe agent.

Classe client : Elle contient les informations sur les clients du système.

Classe profile : Elle contient les informations sur les profiles des clients du système.

Classe agent fournisseur : Elle contient les informations sur les agents fournisseurs de services dans le système. Elle hérite de la classe agent.

Classe fournisseur : Elle contient les informations sur tous les fournisseurs des services dans le système.

5.2. DB_MP (la base de données au niveau de la MP) :

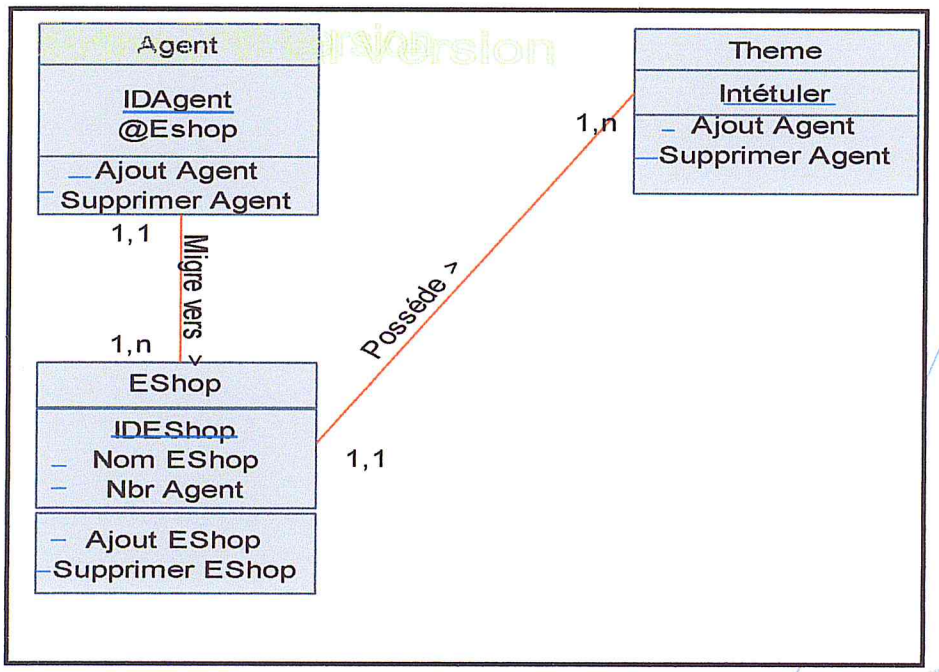


Figure2.5.1 : diagramme de classe sur la BD-MP

Classe Boutique : Elle contient les informations sur les boutiques d'une MP donnée.

Classe agent : Elle contient les informations sur tous les agents (chercheurs et vendeurs) qui résident au niveau d'une MP.

Classe thème : Elle contient les informations sur tous les thèmes de recherche (services) fournis au niveau d'une MP.

5.3. B_Connaissance:

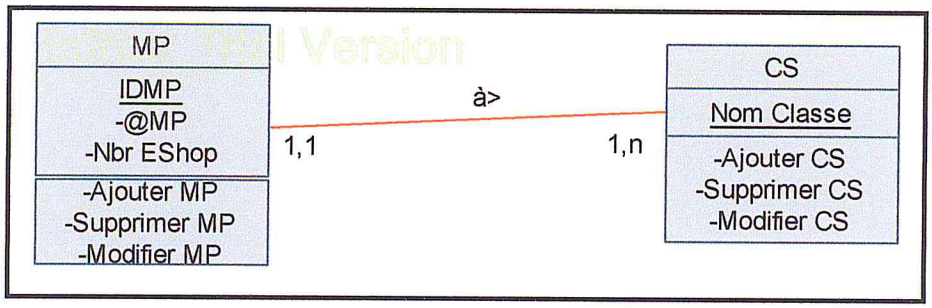


Figure2.5.2 : diagramme de classe sur la B-Connaissance

Classe MP : Elle contient les informations sur toutes les MPs du système.

Classe CS : Elle contient les informations sur toutes les classes de services hébergées dans le système.

5.4. DB_TSA (la base de données au niveau de la TSA) :

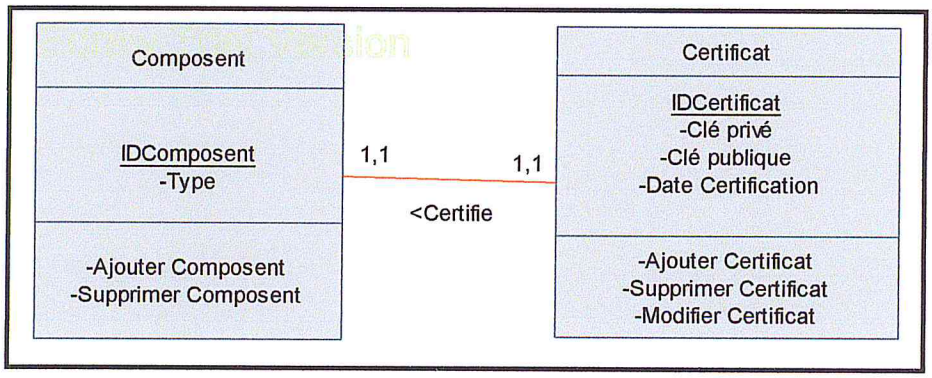


Figure2.5.3 : diagramme de classe sur la BD-TSA

Classe composant : Elle contient les informations sur toutes les entités du système

Classe certificat: Elle contient les informations sur la certification des composants du système.

5.5. BD-boutique (la base de données au niveau des Boutiques):

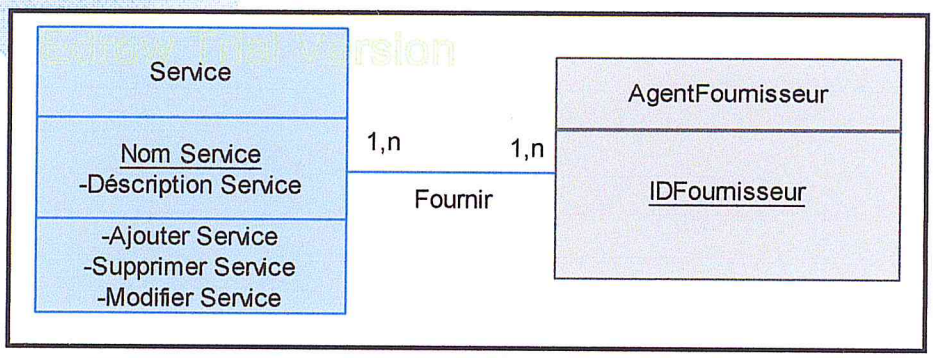


Figure2.5.5 : diagramme de classe sur la BD-boutique

Classe service: Elle contient les informations sur les services fournis par un agent fournisseur.

Classe agent fournisseur : Elle contient les informations sur les agents fournisseurs des services dans une e-shop donnée.

Conclusion :

Dans cette partie on a présenté une nouvelle architecture (IMAN) pour la RI, qui règle les problèmes de sécurité existants sur les SRI par agent mobile Classique.

Partie III:

Chapitre 5:

Implémentation

1. Introduction :

Dans le but d'évaluer le modèle SB, nous reprenons l'exemple de l'application de collecte de documents pour réaliser des prototypes. Nous réaliserons deux prototypes de systèmes de recherche de documents à d'agents mobiles classiques et système buyer-seller sous jade. Qui ont étés tester sous des machines virtuelles (2 machines). Le langage de programmation utilisé est **JAVA** comptabilisé avec la plateforme **JADE** utilisé pour l'écriture des agents.

Dans cette partie on va parler de la conception du système jade sur notre système IMAN puis l'architecture matérielle de notre système qui va décrire les composantes et les interactions entre ces derniers puis l'architecture logiciel qui va définir le déroulement de nos agents sur la plate forme

1. Technologies utilisées :

Nous allons présenter les technologies et les outils utilisés durant cette phase.

1.1. Plateforme 

Jade est une plateforme gratuite et Open Source pour le développement des systèmes à base d'agent. Développée au sein des laboratoires de «*Telecom Italia*», cette plateforme facilite le développement des systèmes à base d'agents répondant aux spécifications **FIPA**.

Le choix de la plateforme JADE :

La plupart des plateformes d'agents mobiles sont dépassé par le temps. Aglet et JavAct restent des plateformes intéressantes mais les points forts de Jade par rapport à ces deux plateformes sont la richesse et la puissance de son API d'une part, et la forte compatibilité FIPA d'autre part.

Nous avons utilisé dans notre système la version 3.5 de JADE téléchargeable sur le site officielle du. La version la plus récente est la 3.6.1.

1.2. IPMS (Intra Platform Mobility Service):

IPMS (Intra Platform Mobility Service) est un *add-on* pour Jade conçu spécialement pour la migration des agents entre plusieurs plateformes

1.3. Les agents de JADE:

Jade compte quelques agents de base indispensables pour le bon fonctionnement de la plateforme. Ces agents sont :

Nom de l'agent	Signification	Description	Indispensable pour le lancement de JADE
AMS	Agent Management System	gère l'ensemble de la plateforme	**Oui**
DF	Directory Facilitator	représente la mémoire de la plateforme	**Oui**
RMA	Remote Management Agent	fournit une interface graphique pour la gestion de la plateforme	**Non**

1.4. Les Containeurs:

Les agents Jade s'exécutent dans des environnements spécifiques appelés **Container** (conteneur). Un ensemble de container forment une **plateforme**. Chaque plateforme dispose d'un seul et unique Container spéciale appelé **Main-Container** qui représente toute la plateforme et auprès duquel tous les autres containers doivent s'inscrire. Ce container a la particularité d'héberger trois agents spéciaux : AMS, DF et RMA.

L'ensemble des Containers peut être distribué sur plusieurs machines d'un réseau local. La figure suivante montre un réseau qui contient deux plateformes Jade, l'une avec trois containers et l'autre avec un seul container :

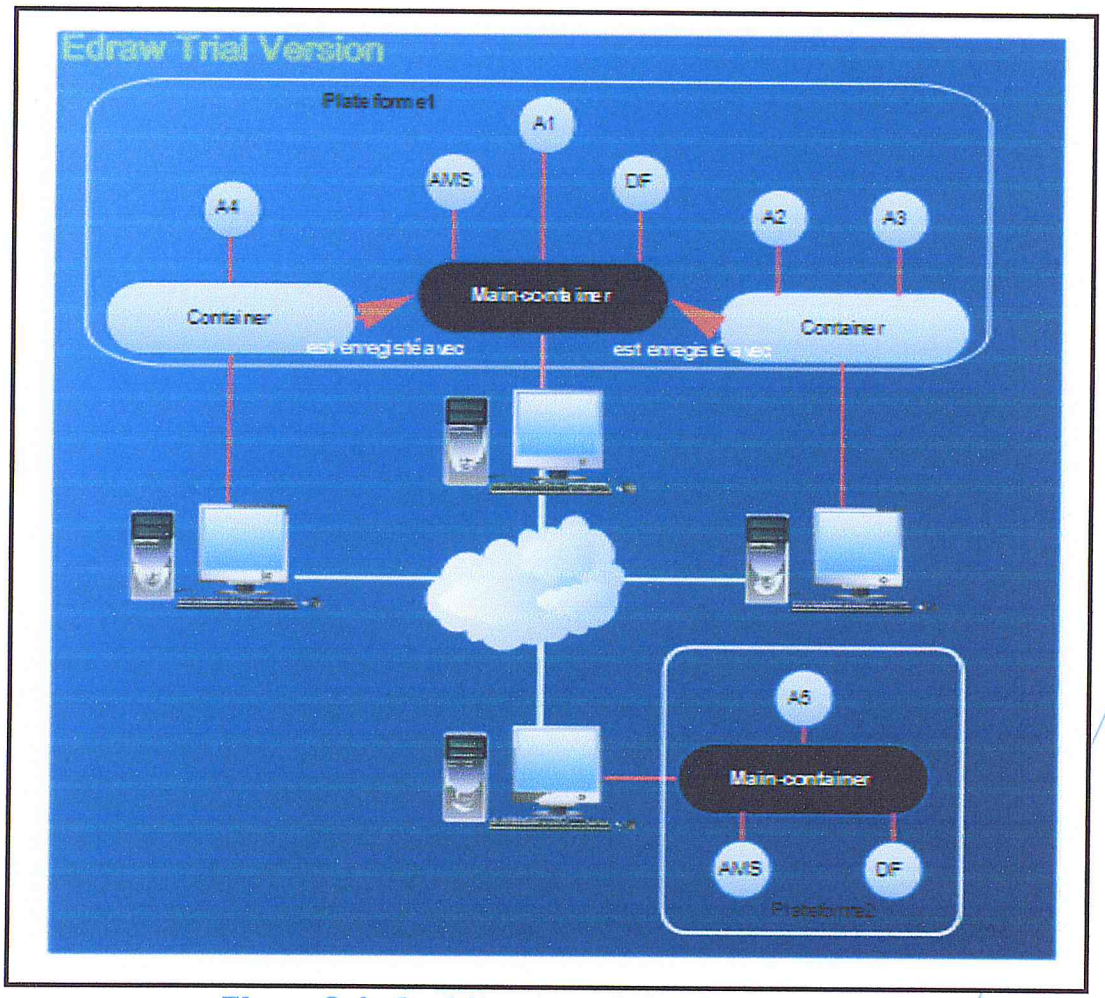


Figure3.1: Architecture JADE distribuée

1.5. La mobilité de JADE:

Dans sa version de base, Jade fournit un service de mobilité permettant aux agents de migrer d'un container à un autre sur la *même* plateforme. D'un point de vue implémentation, cette mobilité ce fait grâce à la méthode suivante :

doMove(Location destination)

Où Location est une interface (au sens objet du terme) implémentée par la classe ***ContainerID*** (une instance de cette classe représente un Container).

Dans la version de base, il n'est pas possible pour un agent que de migrer d'un container à un autre sur la même plateforme (et forcément sur le même réseau local). Pour réaliser une migration entre deux plateformes, par exemple entre Platform1 et Platform2 dans la figure 6.1, il faut utiliser l'add-on **IPMS**. La migration entre les plateformes avec IMPS se fait par la méthode *doMove()* précédente, mais le paramètre destination devient une instance de la classe ***PlateformID*** .JADE offre trois types de mobilité :

Type	Nombre de plateformes	Type de réseau	Type
Type I	1	Sur une seule machine	Type I
Type II	1	Réseau local	Type II
Type III	Plusieurs	Réseau étendu	Type III

2. Adaptation de JADE:

Au court de la programmation, il nous faut quelques adaptations de la plateforme JADE de base:

2.1. Modification du TimeOut:

Sur l'IMPS l'add-on le ***timeOut*** de la migration à 30 secondes. Au-delà de cette période, si un agent n'arrive pas à achever sa migration, l'opération sera annulée. Comme les agents de notre système doivent transporter plusieurs fichiers avec des tailles variées, l'opération de sérialisation et de la désérialisation de l'agent prend un temps proportionnel à la taille des fichiers transportés. Dans les premiers tests, il n'était pas possible de dépasser les 50 ko par agent. Il a fallu donc modifier le code de l'add-on IMPS.

Pour cela, nous avons augmenté le timeout au fur et à mesure du besoin pendant les tests jusqu'à arriver à 60 minutes. La modification se fait au niveau de la classe **InterPlatformMobilityService.java** du paquetage **jade.core.migration** au niveau des deux constantes :

```
public static final String MESSAGE_RESPONSE_TIMEOUT = "3600000";
public static final String MESSAGE_RESPONSE_TIMEOUT_RESPONDER = "3600000";
```

Respectivement ligne 108 et ligne du code 109.

La modification apportée à l'IPMS a permis aux agents de transporter jusqu'à 100 fichiers de 20 ko chacun. Cependant, cela a provoqué un nouveau problème au niveau de la *java virutal machine*, du fait que la mémoire allouée à la *JVM* s'épuise dès que la taille de l'agent et ses fichiers dépasse un certain seuil. La solution est de mentionner à la *JVM* la taille qu'elle doit allouer et cela par la commande suivante :

```
java -Xms256m -Xmx512m -XX:PermSize=64M -XX:MaxPermSize=1000M
```

3. Architecture JADE du système:

Dans ce qui suit, nous présentons comment réaliser l'architecture MP avec des plateformes Jade.

3.1. Vue générale du système

Le système se compose d'un ensemble de plateformes Jade réparties sur plusieurs machines. Une plateforme Jade implémente une **place de marché MP**. Chaque **boutique** (ou eshop) est représentée par un container (autre que main-container).

Les containers des boutiques de la même MP doivent se connecter au Main-container de la MP. Sur chaque machine MP, une instance de la plateforme Jade s'exécute. Ainsi, chaque MP est représentée par une instance de la plateforme Jade (donc un main-container) à laquelle sont connectés les containers des différentes boutiques.

3.1.1. Les places de marché MP

Une MP est constituée d'un ensemble de machines interconnectées. Cet ensemble contient une machine principale. Cette machine est la machine sur laquelle l'instance de la plateforme Jade de la MP s'exécute. Sur les autres machines (ou machines secondaires), s'exécutent des Containers qui sont tous connectés au main-container de la machine principale.

Les agents facilitateurs du système sont représentés par l'agent **DF** de la plateforme qui représente la mémoire de la MP. Pour la persistance des données de l'agent **DF**, nous utilisons une base de données MySQL, recommandé pour l'utilisation avec Jade. La figure suivante montre le schéma d'une MP sous Jade.

3.1.2. Les boutiques

Dans une boutique, c'est-à-dire dans un conteneur, nous avons besoin d'un agent qui gère et manipule la base de données de la boutique. Cet agent doit avoir les mêmes caractéristiques que l'agent **DF**. Comme on ne peut avoir qu'un seul **agent DF** dans toute la plateforme, Jade offre un mécanisme pour créer des agents qui ont les mêmes caractéristiques avec le DF.

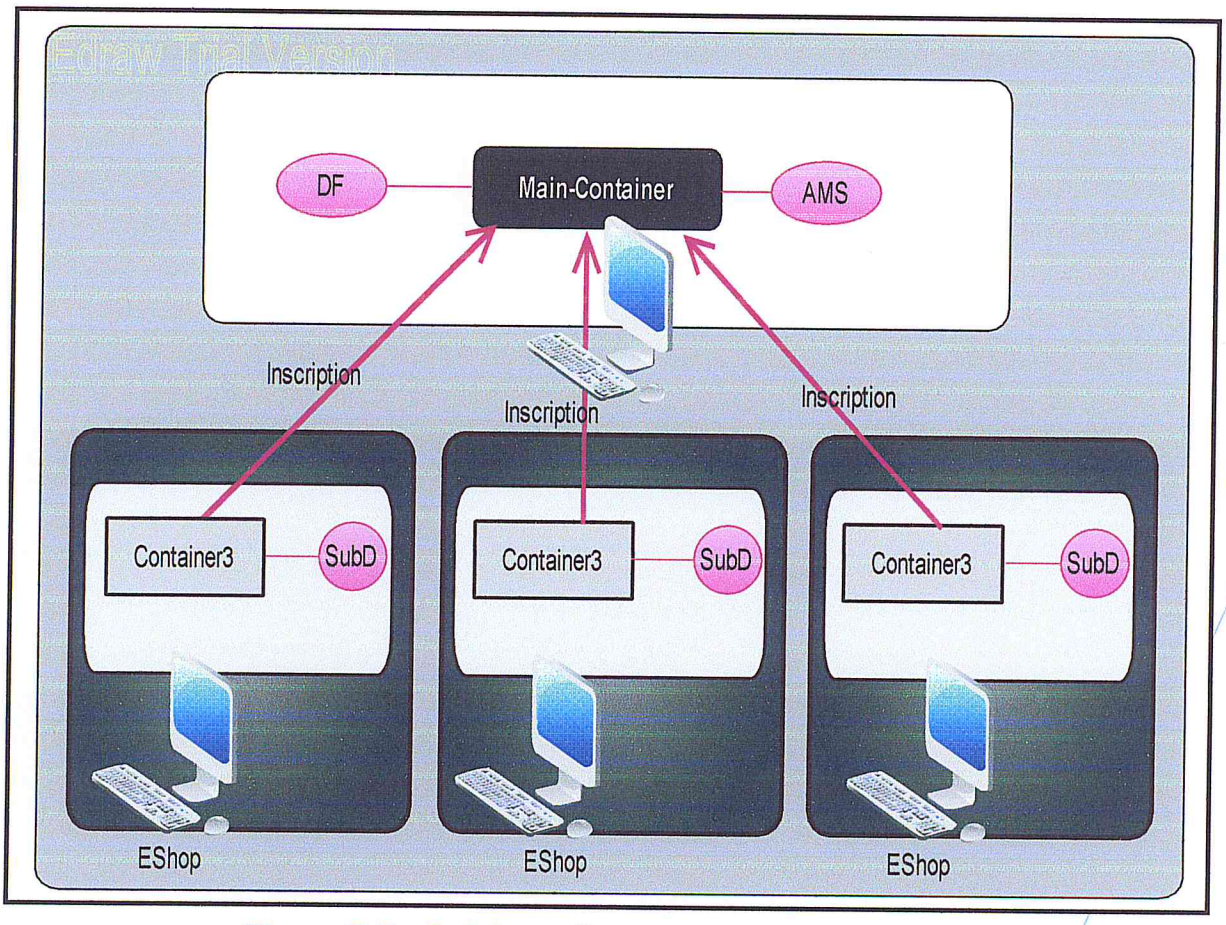


Figure3.2 : Schéma d'une MP sous JADE

3.1.3. Les sites ASP

L'**ASP** est un site sur lequel s'exécute une instance Jade. Cette instance va permettre de créer les agents mobiles dans le système. L'ASP et grâce a son **IHM** qui permet à l'utilisateur de saisir une **requête** pour rechercher un ensemble de documents.

3.2. Les agents du système :

Le système compte trois types d'agents :

- agents chercheurs de documents.
- agents fournisseurs de documents.
- Agents de services.

Ces trois types d'agents héritent tous de la classe *Agent* du paquetage *jade.core.Agent*. Chaque type diffère des deux autres par la manière d'implémentation des méthodes de base des agents, la logique suivie par les agents et leurs comportements.

Les tâches de chaque agent Jade sont appelées **Behaviours** ou comportements. Une fois un behaviour créé, il est automatiquement planifié pour être exécuté.

3.2.1. Les agents chercheurs :

Les agents chercheurs ont la particularité d'avoir un modèle d'autonomie de décision assez avancé par rapport aux deux autres types d'agents car il combine plusieurs types de behaviours.

3.2.2. Les agents fournisseurs:

Les agents fournisseurs implémentent également des automates à états finis mais de moindre complexité que ceux des agents chercheurs. Ceci est dû à l'absence de migration multiple de ces agents (ils tendent à demeurer dans les boutiques jusqu'à épuisement ou obsolescence du service transporté).

3.2.3. Les agents de services

Les agents de services du système sont les agents qui fournissent des services complémentaires au système. **C'est des agents** facilitateurs **qui fournies les** itinéraires aux agents sur les différents sites du système.

3.3. La négociation entre les agents

La négociation entre les agents fournisseurs et les agents chercheurs du système se fait au niveau des boutiques. Cette négociation ce fait selon un protocole de négociation appelé **FIPA Contract-Net.** Le FIPA Contract-net comporte deux acteurs : l'**initiateur** de la négociation et les **participants**.

L'initiateur est l'agent chercheur qui demande à l'agent boutique de lui fournir la liste de tous les fournisseurs du service qu'il cherche. La négociation est lancée par l'agent chercheur qui fait un **appel d'offre** par envoi de message de type *CFP* (Call For Porposal) à tous les agents fournisseurs retournés par le Sous-DF. Les agents qui reçoivent cet appel d'offre peuvent répondre par un message de type *INFORM* qui contient l'un des messages suivants :

- **NOT-UNDERSTOOD** : dans le cas où l'appel d'offre n'est pas supporté par l'agent.
- **REFUSED** : dans le cas où l'agent fournisseurs ne veut pas répondre à l'appel d'offre.
- **PROPOSE** : dans le cas où l'agent est intéressé par l'appel d'offre.

L'agent chercheur envoie les **mots clé,** constituant la requête de recherche, au sous-ensemble d'agents fournisseurs qui ont offert leurs services. Les agents fournisseurs consultent leur base d'index et passent les liens (URL par exemple) des fichiers pertinents dans un message à l'agent chercheur.

4. Architecture matérielle du système :

Cette architecture présente l'hiérarchie du système qui constitue des machines interconnecté qui sont doté de la plateforme jade, dans le schéma qui va suivre on présente l'architecture matérielle du système a base du prototype IMAN :

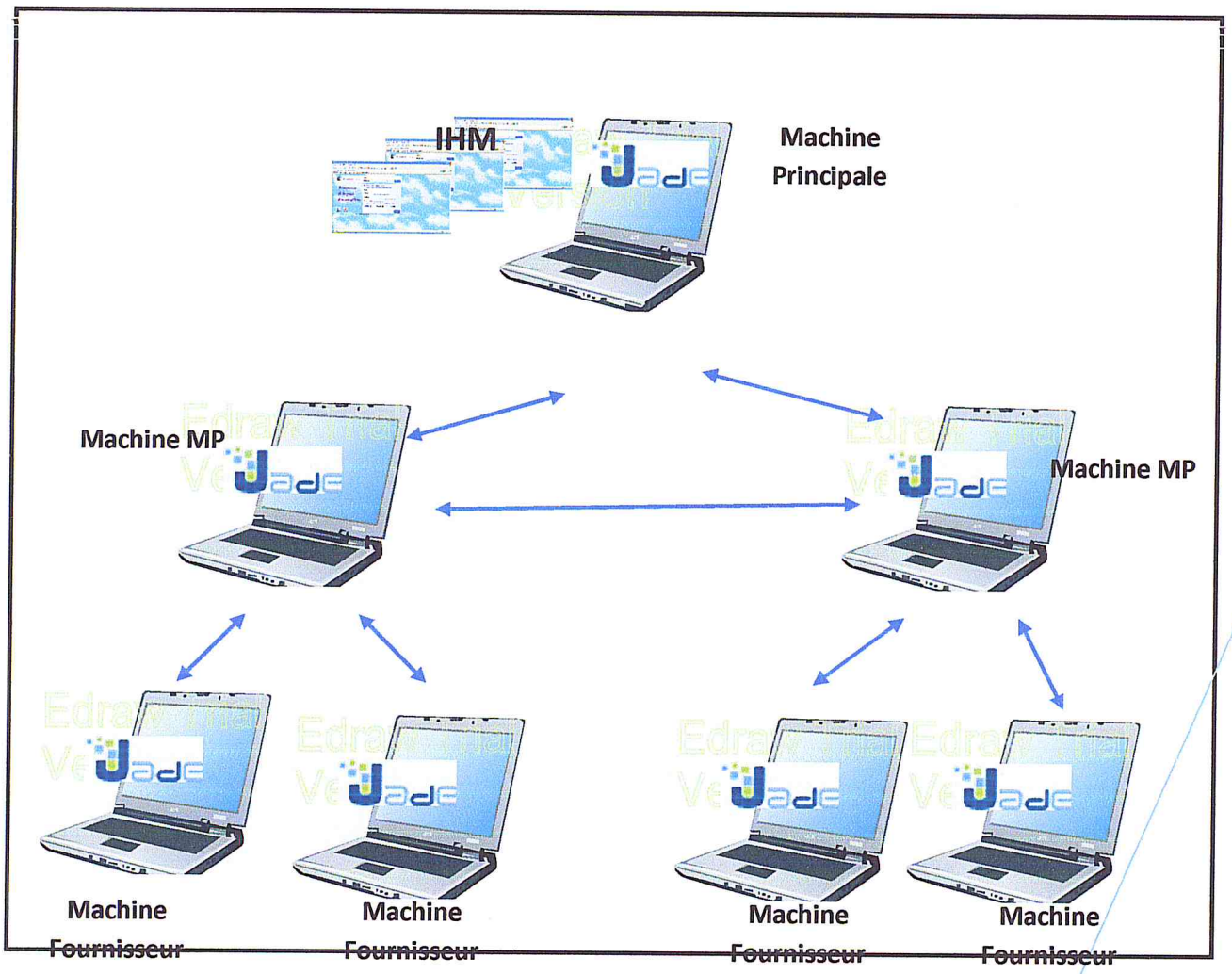


Figure 3.3 : Architecture matérielle du système IMAN

5. Exemples d'implémentations :

Afin d'étudier les performances du système en consommation de la bande passante en a tenter de présenter un système de recherche de base (simple).

5.1. Fonctionnement d'un SRI de base:

L'architecture est présentée comme un ensemble de machines interconnectés qui s'échange de l'information à travers le dialogue avec messages

Se système se compose d'une machine qui comporte un module appelé **URL Server** qui comporte tout les URLs des documents existant qui sont distribuer sur tous les autre machines

Chaque fois la machine initiale cherche : le document il est retourné a travers le module **doc index** qui contient les indexes sur les documents ainsi que **store server** ou se trouve le contenue du document

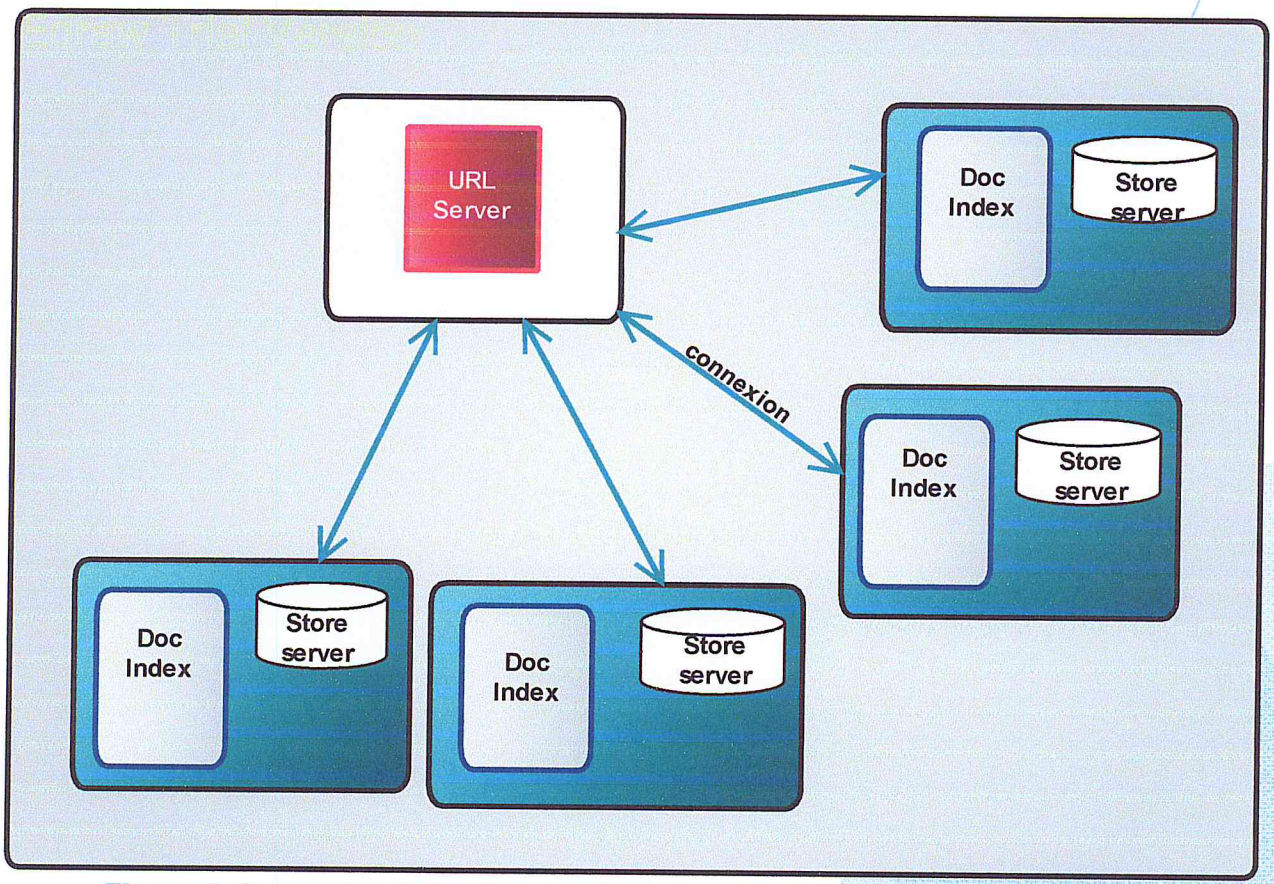


Figure 3.4 : architecture d'un système de recherche d'information de base

5.2.Fonctionnement d'un SRI de base sur l'architecture IMAN :

Le système est un ensemble de plateformes interconnecté selon notre architecture IMAN. Sur chaque plateforme on a un environnement d'exécution des agent et c'est le main containeur qui contient un ensemble de conteneur qui vont s'inscrire sur se dernier. Les agents facilitateurs s'inscrivent sur le main-containeur et redirigent les agents vers la bonne destination (boutique destination).

5.2.1. Architecture matérielle:

c'est la même architecture présentée dans la le paragraphe 4 (la figure 3.3)

5.2.2. Architecture logicielle :

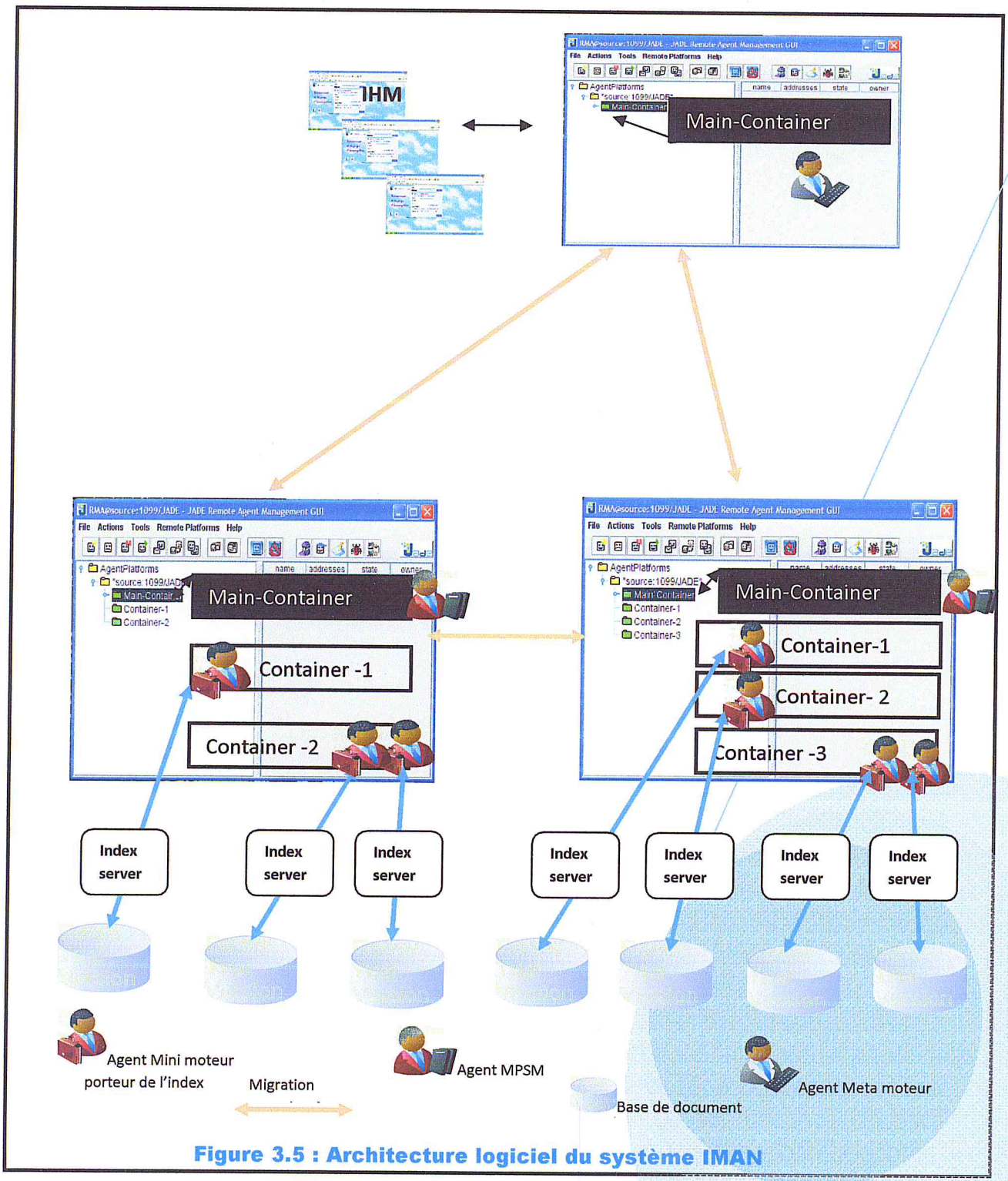


Figure 3.5 : Architecture logiciel du système IMAN

Se système est constituer d'une ensemble de plateformes JADE, qui s'exécute sur le Main-Container, dont les boutiques qui sont représentés par des container doivent-y inscrire

5.3.Fonctionnement d'un SRI simple à base d'agents mobiles classique:

Dans se système les agents représentes des entités qui se constitues de :

- Le code de l'agent
- L'état d'exécution
- Requête de l'agent
- Ensembles des résultats

Cette architecture est constituée d'un ensemble de plateformes JADE, que le main container présente son environnement d'exécution

5.3.1. Architecture logicielle : se système est constitué d'un ensemble de plateformes JADE, qui s'exécute sur le Main-Container, dont les boutiques qui sont représentés par des container doivent-y inscrire, pour l récupération des documents l'agent a besoin de Store server qui contient des descriptions des documents, Doc index qui contient l'intitulé du document la partie cliquable lors de la présentation de la recherche et URL Server qui a tout les adresses IP de tout les sites disponibles.

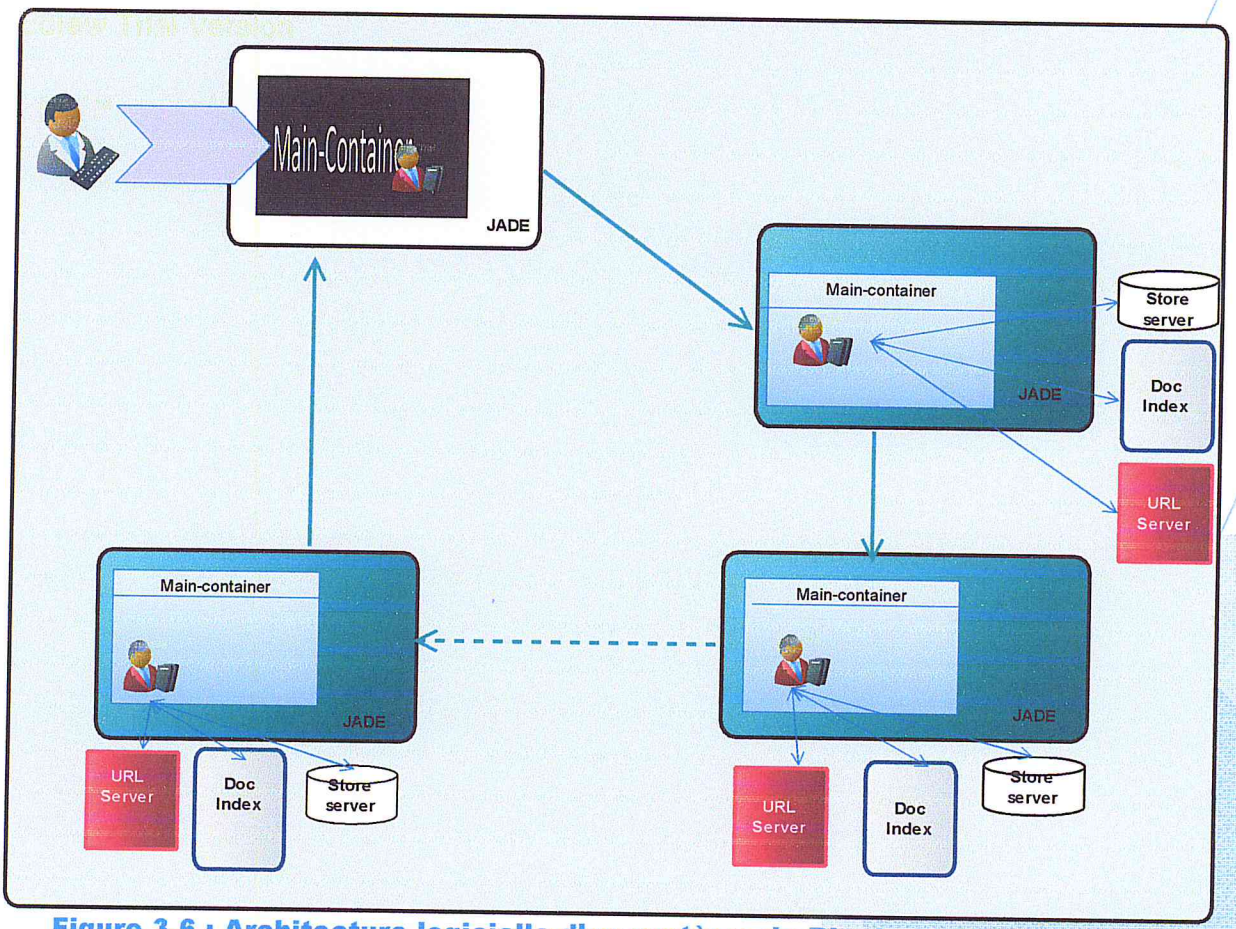


Figure 3.6 : Architecture logicielle d'un système de RI a base d'agent mobile

5.3.1. Architecture matérielle :

Architecture matérielle d'un système de RI a base d'agent mobile contient un ensemble de machines qui sont doté d'une plateforme JADE, sur la machine principale on lance la création de l'agent qui cherche l'information trouvé sur les machines distribuées.

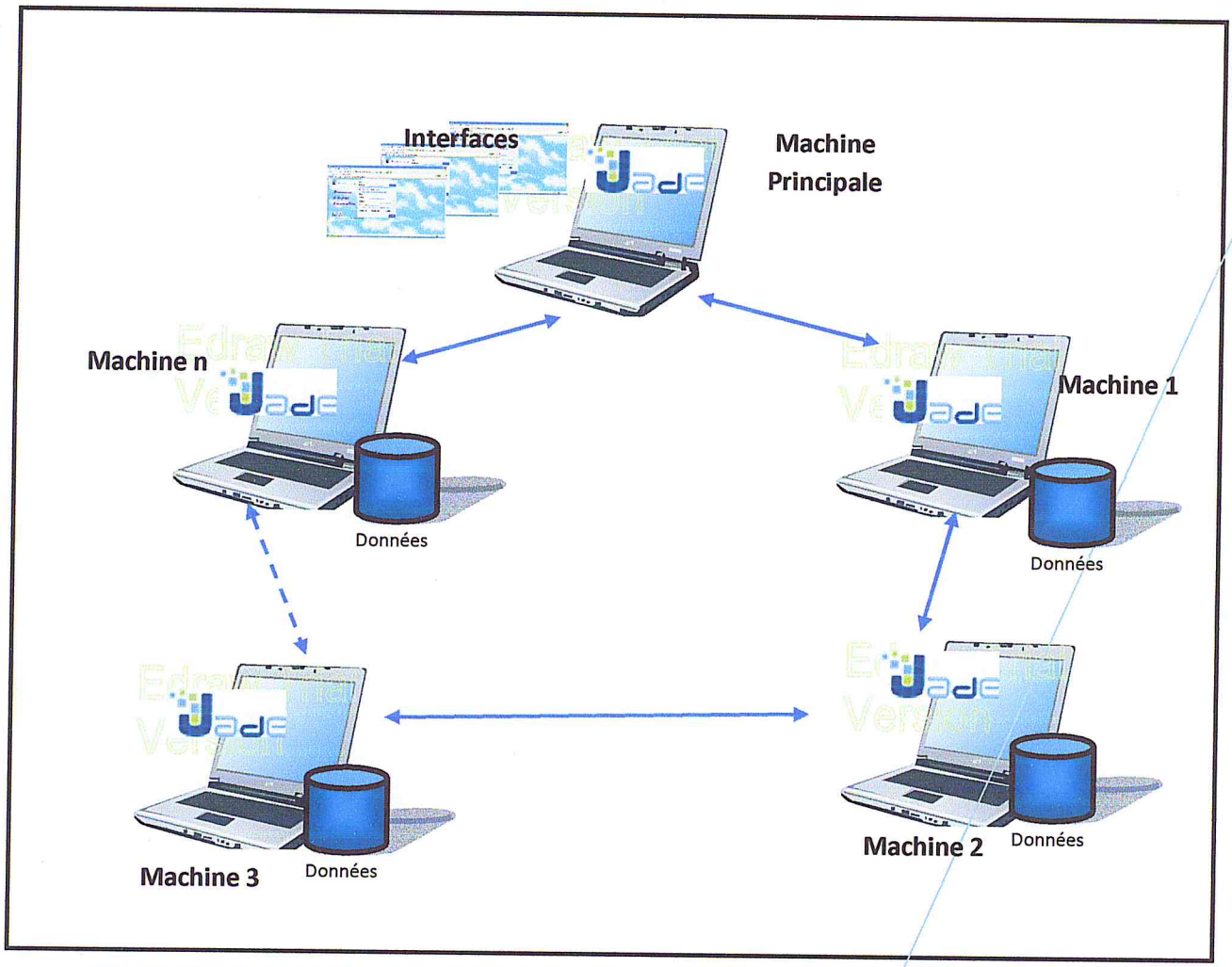
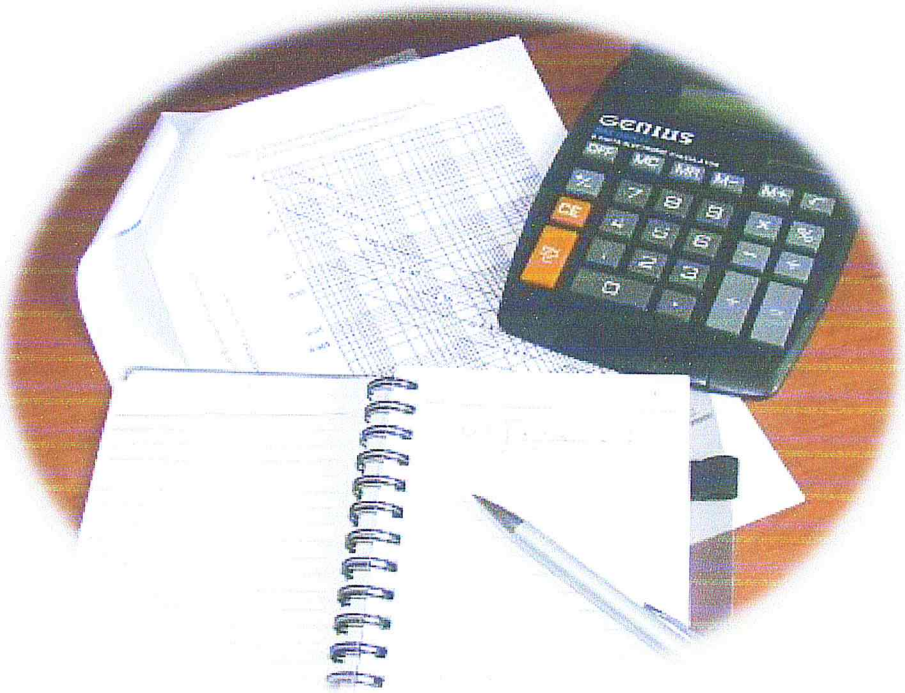


Figure 3.7 : Architecture matérielle d'un système de RI a base d'agent mobile


Genius

1. Introduction

Dans cette dernière partie nous allons étudier la performance des trois deux modèle conçus, et comme en a déjà expliqué dans la partie de la recherche d'information qu'il existe plusieurs critères pour en jugé de la performance d'un SRI, parmi eux la consommation de la bande passante sur la quelle on va se basé pour faire nos tests :

2. Formules dévaluations :

2.1. L'architecture IMAN:

Chaque site fournisseur créé un agent fournisseur qui va le représenté. Cet agent va porter avec lui le fichier index, qui contient tous les entêtes des documents qui se trouvent sur le site fournisseur, puis il se déplace vers la MP qui correspond à sa catégorie de service ou :

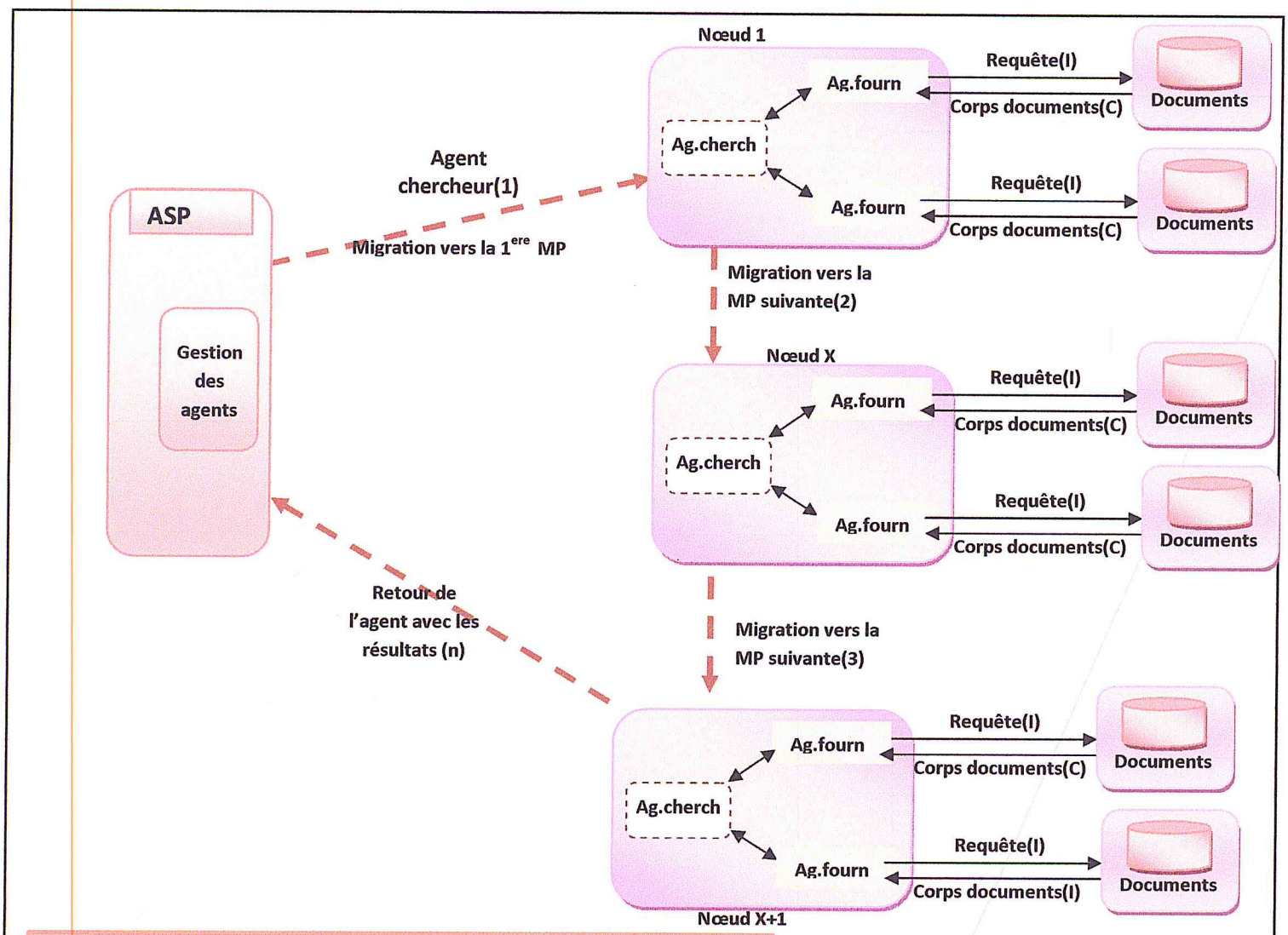
- Chaque fournisseur est représenté par un seul agent, et un agent ne représente qu'un seul fournisseur.
- Le système comprend n agents fournisseurs par MP, et m MP au totale.

Après la création de l'agent chercheur, se déplace de l'ASP vers la première MP de son itinéraire, donc le trafique initiale de la machine du client est :

$$T_{Site\_client} = A + I$$

A son arrivé à la MP, l'agent chercheur communique localement avec les agents fournisseurs (Négociation Contract-net), ensuite, il leurs demande un ensemble de fichiers (résultats de la négociation). Chaque agent fournisseur télécharge localement les fichiers qui ont lui été demandés. Pour le faire, il doit faire une requête de taille I pour chaque document (le corps du document de taille C), chaque site fournisseurs contient iD documents pertinents. Donc l'agent va faire iD requêtes et télécharger iD documents.

$$T_{agent\_fournisseur} = iD * I + iD * C$$



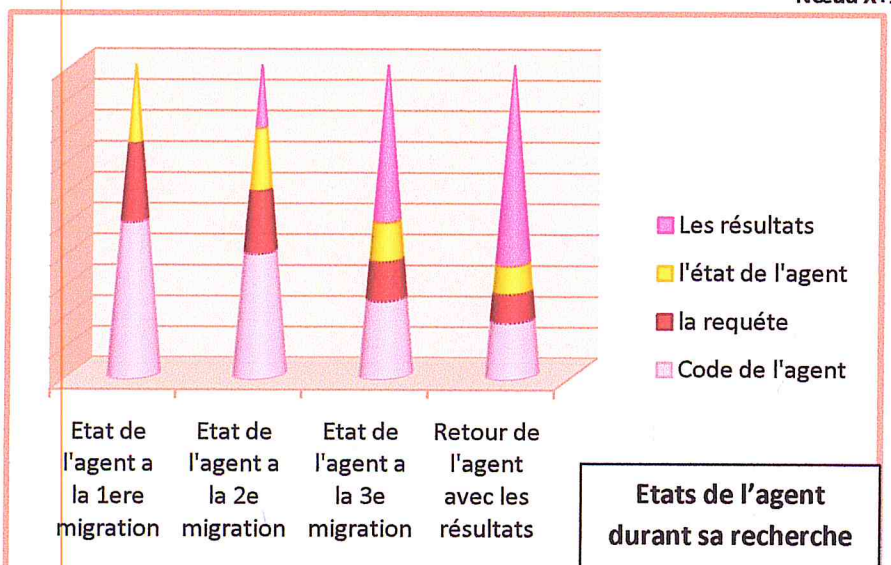


Figure 1.1: Architecture recherche d'information simplifiée pour l'architecture IMAN

2.2. Le cas des agents mobiles classiques :

Dans le cas des agents mobiles classiques, le browser se déplace vers chaque nœud. Ensuite, il consulte les données de chaque nœud pour enfin repartir avec un ensemble de documents pertinents et une mise à jour de l'URL server. A chaque saut d'un nœud à un autre le browser prend avec lui son code, son état d'exécution et l'ensemble de documents trouvés.

Le trafic sera alors :

$$T = G + S^j + r$$

Ou :

- G est le code de l'agent.
- S^j la taille de l'état de l'agent.

Le S^j se calcule de la manière suivante :

$$S^j = d_{sa} + S + \sum_{j=0}^{N} iDb$$

Ou :

- d_{sa} est la taille de la liste des URL trouvé sur l'URL server
- S la taille du l'état de l'agent.
- $\sum_{j=0}^{N} iDb$ est la taille des fichiers récolté au cours du parcours de l'agent jusqu'au nœud j.

Pour simplifier la formule on pose : $\bar{S} = d_{sa} + S$

Ce qui devient :

$$T = \sum_{j=0}^{N}\left(I + G + \bar{S} + \sum_{1}^{j} iDb\right) = \left(I + G + \bar{S} + \frac{N}{2} iDb\right)(N+1)$$

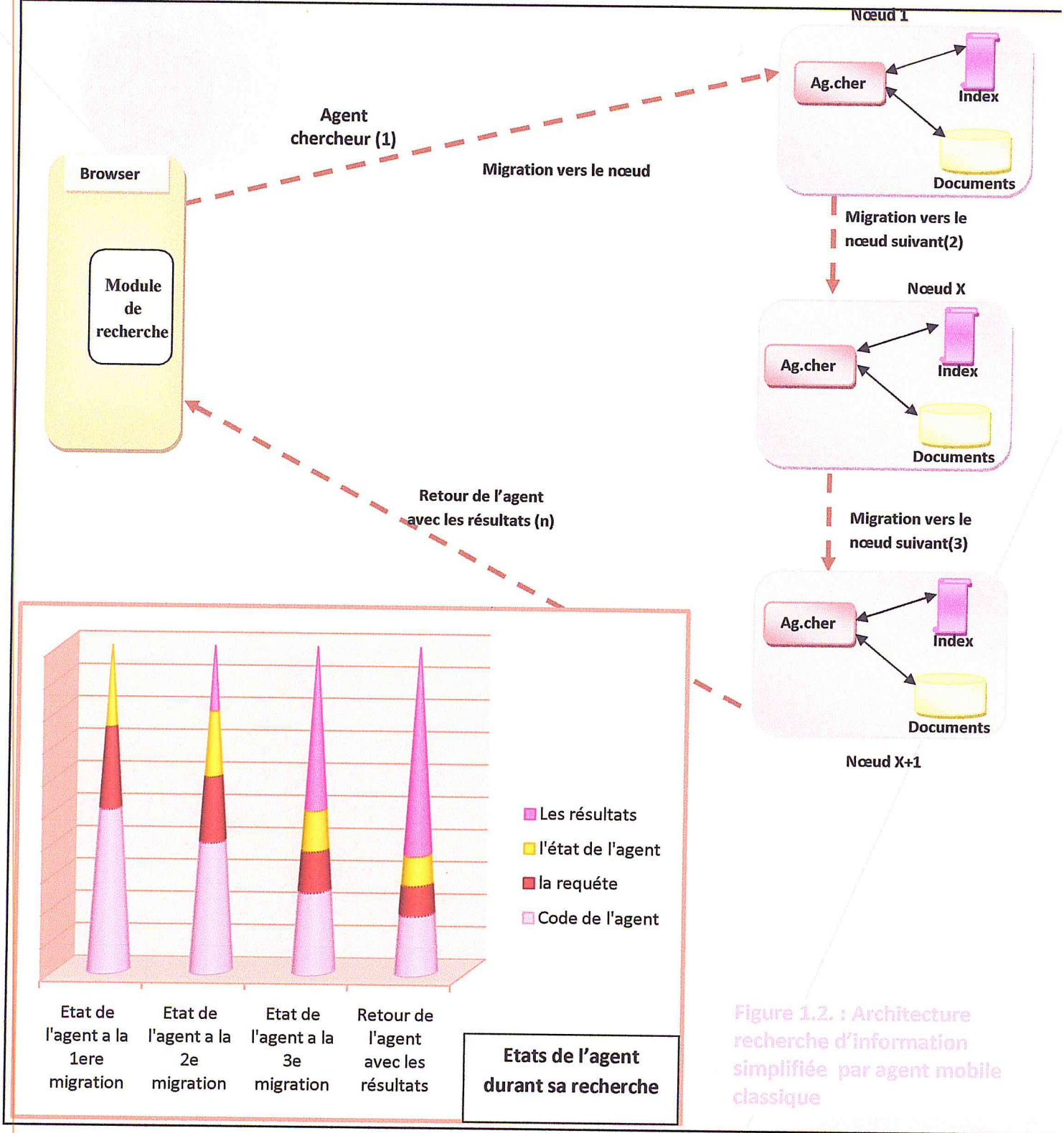


Figure 1.2. : Architecture recherche d'information simplifiée par agent mobile classique

Ces résultats nous permettent de déduire :

- Le modèle SB (Seller-Buyer) est meilleur que le modèle MA.
- Dans le premier cas le SB1 (à une place de marché) est meilleur que les modèles MA et SB2 (à deux places de marché).
- A partir d'un nombre de 5 fournisseurs et plus le modèle MA devient mauvais.

Il ressort que le SB devient progressivement meilleur en augmentant le nombre des MP par rapport au modèle AM classique. Cette augmentation devient très visible à partir d'un nombre 3MP. Cette augmentation peut être déterminée en calculant un seuil de satisfaction comme suit :

Satisfaction = Nombre de fournisseur / nombre MP

Cette formule est meilleur dés que les MP= 3 c'est-à-dire la satisfaction est égale a 1,66

Conclusion :

On peut conclure que notre modèle d'agent mobile est toujours meilleur par rapport au autres modèle agent mobile existant mais aussi le système SB donne des donne de meilleur performance que le MA si la satisfaction atteint un certain seuil. Néanmoins le client serveur consomme moins de la bande passante que les deux autres modèles car ce modèle échange un minimum de requête/réponse par contre les agents mobiles transportent avec eux les fichiers d'une machine à une autre.

Conclusion Générale

A travers ce projet nous avons proposé une solution ou une partie de solutions à trouver dans le domaine de la sécurité de la recherche d'informations basée sur les agents mobiles.

Notre modèle d'architecture conçu dans ce travail pour donner un maximum de sécurité aux agents mobiles nous a permis de proposer un site de recherche d'informations **hautement sécurisé**.

Notre modèle d'architecture a été basé sur :

- L'authentification des utilisateurs du système ;
- La création d'un seul agent mobile pour une seul recherche ou service ;
- Utilisation des agents facilitateurs dans la garantie de protection de l'information ainsi que des agents ;
- L'utilisation des mises à jour ;

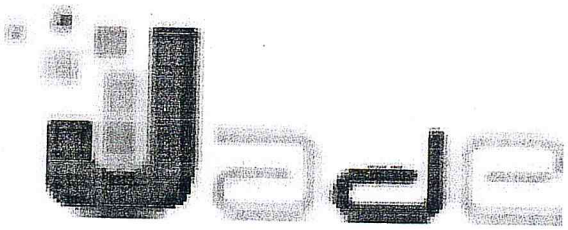
Afin de montrer l'apport de ce modèle dans le domaine de recherche d'information, nous avons développé un prototype de système de recherche d'information basé sur l'interaction et la coopération d'un ensemble d'agents mobiles.

Ce site est destiné à combler les points suivants :

- Permet à un utilisateur fournisseur de déposer ses services ;
- Permet à un utilisateur client de trouver l'information désirée et d'y accédée en toute sécurité ;
- Permet de diminuer le temps de recherche grâce à l'utilisation de plusieurs agents donc diminuer le temps de connexion ;
- Permet à un client de voir son résultat à n'importe qu'elle moment sans qu'il soit être obligé de rester connecté (mode déconnecté).
- Permet à un client d'avoir un résultat varié et riche grâce à la visite de plusieurs places de marché, boutiques.

Afin d'élargir les fonctionnalités du système, nous proposons comme perspective une réalisation complète de l'architecture proposée car nous avons considéré dans notre application une seule place de marché qui est elle-même une boutique, ce qui élimine la migration de l'agent mobile d'une place de marché à une autre et d'une boutique à une autre ainsi que la variété et la richesse de l'information trouvée. Il s'agira aussi de tester notre système sur d'autre domaine tel que le commerce électronique et les enchères électroniques.

Annexe1 : La plate-forme



Introduction :

La plate-forme JADE (Java Agent DEvelopment Framework) est un environnement de développement d'agents implanté totalement dans le langage JAVA. Il facilite la mise en place d'un système multi agents répondant aux spécifications de FIPA (Foundation for Intelligent Physical Agent) à travers un ensemble d'outils.

1. Présentation

1. 1. Généralités

JADE (**J**ava **A**gent **DE**velopment framework) est une plate-forme Multi-Agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie). Elle a pour but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme FIPA (FIPA, 1997). JADE comprend deux composantes de base :

- Une plate-forme agents compatibles FIPA
- un paquet logiciel pour le développement des agents Java. JADE inclut :
  1. un environnement d'exécution **(runtime environment)** où les agents JADE vivent ;
  2. une bibliothèque des classes que les programmeurs utilisent (directement ou en les spécialisant) pour développer leurs agents;
  3. une suite d'outils graphiques qui permet d'administrer et de surveiller l'activité courante des agents;

Chaque instance de l'environnement d'exécution de JADE s'appelle un conteneur (**container**) car elle peut contenir plusieurs agents. L'ensemble des conteneurs actifs s'appelle une plateforme. Dans une plateforme, un conteneur spécial appelé conteneur principal (Main-container) doit toujours être en activité, les autres conteneurs s'enregistrent auprès de celui-ci dès qu'ils démarrent.

La figure suivante illustre un exemple de deux plateformes composées respectivement d'un et de trois conteneurs. Les agents JADE sont identifiés par un nom unique ce qui leur permet de communiquer d'une manière transparente indépendamment de leur localisation : même conteneur (par exemple les agents A2 et A3 sur la figure),

des conteneurs différents dans la même plateforme (par exemple A1 et A2) ou des plateformes différentes (par exemple A4 et A5).

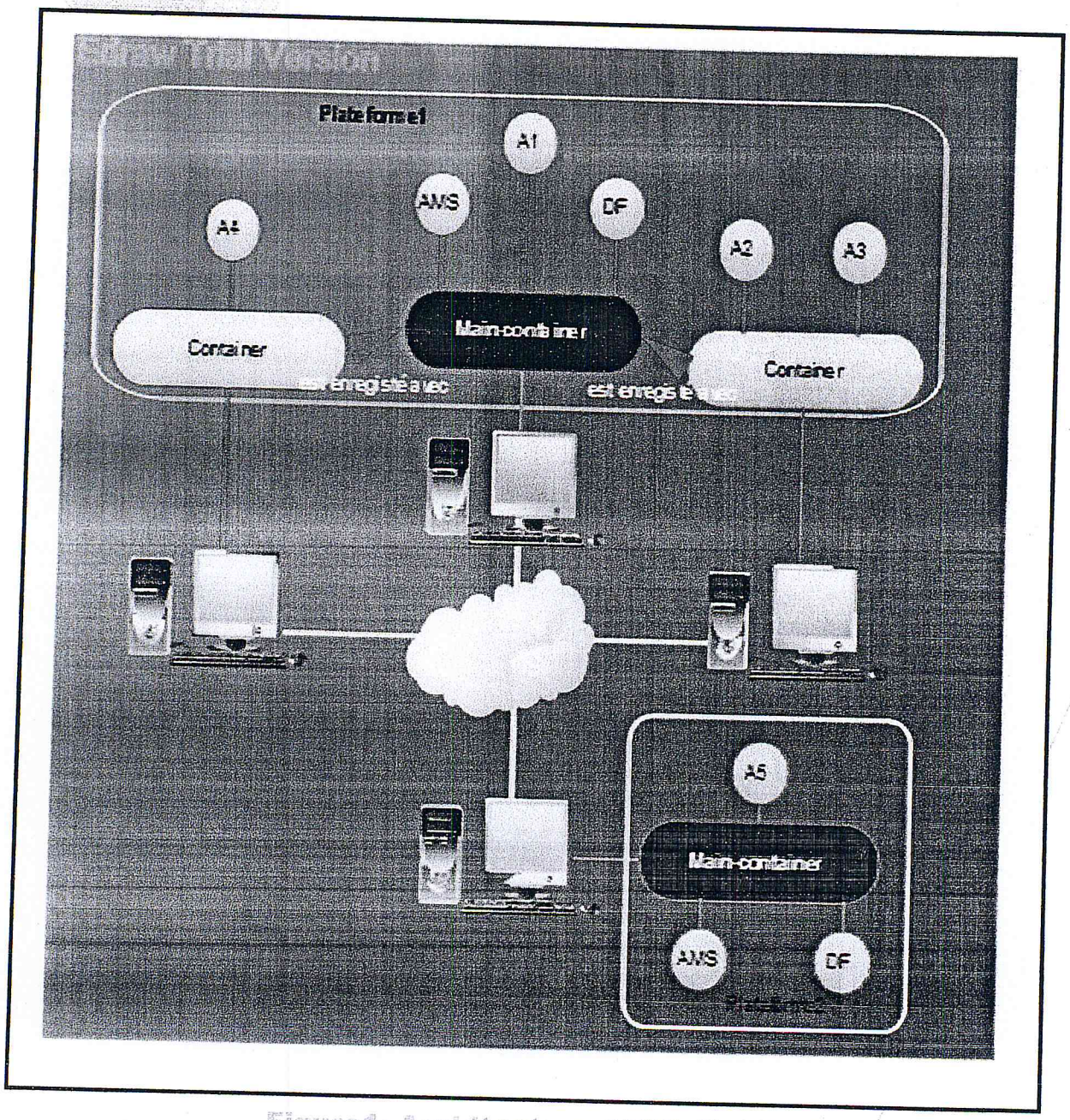


Figure1: Architecture JADE distribuée

La plate-forme d'agents peut être répartie sur plusieurs serveurs. Une seule application Java est exécutée sur chaque serveur. Chaque JVM (machine virtuelle Java) est un conteneur d'agents qui fournit un environnement complet pour l'exécution des agents et permet à plusieurs agents de s'exécuter en parallèle sur le même serveur.

La plate-forme Multi-Agents JADE est alors composée de plusieurs conteneurs d'agents. La distribution de ces conteneurs à travers un réseau d'ordinateurs est permise, à condition que la communication RMI entre leurs hôtes soit conservée.

Chaque conteneur d'agents est un objet serveur RMI qui gère localement un ensemble d'agents. Il règle le cycle de vie des agents en les créant, les suspendant, les reprenant et les détruisant. En plus, il traite tous les aspects de la communication :

- répartition des messages ACL reçus,
- routage des messages selon le champ de destination
- dépôt des messages dans les files de messages privées des agents.

Pour les messages vers l'extérieur, le conteneur d'agents maintient assez d'informations pour chercher l'emplacement de l'agent récepteur et pour choisir une méthode de transport convenable pour expédier le message ACL.

La plate-forme offre une interface graphique utilisateur (GUI) pour la gestion à distance qui permet de contrôler et superviser les états des agents, par exemple arrêter et remettre en marche un agent. L'interface graphique permet aussi de créer et de commencer l'exécution d'un agent sur un hôte éloigné, à condition qu'un conteneur d'agents s'exécute déjà sur cet hôte. L'interface elle-même a été implémentée comme un agent appelé RMA. Toute la communication entre les agents et l'interface (GUI) et toute la communication entre cette interface et l'AMS est faite par ACL via une extension ad hoc de l'ontologie des agents de gestion FIPA.

Jade supporte la gestion des comportements coopératifs. Le run-time inclut quelques fonctions complexes prêtes à l'emploi pour les tâches les plus communes dans la programmation agent, comme des protocoles d'interaction de FIPA.

1. 2. Composants

JADE est composé des principaux packages suivants :

- **Jade.core** implante le noyau du système. Il possède la classe *agent* qui doit être étendue par les applications des programmeurs. Une classe **behaviour** *(comportement)* est contenue dans **jade.core.behaviours**, une sous classe de **jade.core**. Les comportements implémentent les tâches ou intentions des agents.
- **jade.lang** contient un sous-package pour chaque langage de communication utilisé par JADE en particulier **jade.lang.acl** .
- **Jade.content** contient un ensemble de classes qui permettent de définir des ontologies.
- **Jade.domain** contient toutes les classes Java qui représentent les entités Agent Managment définies par FIPA en particulier AMS et DF.
- **Jade.gui** contient un ensemble générique de classes utiles pour la création de GUIs pour l'affichage, l'édition des messages ACL, et de la description des agents.
- **Jade.mtp** contient une interface Java que chaque MTP (Message Transport Protocol) doit implémenter.
- **Jade.proto** contient des classes qui modélisent les protocoles standards d'interaction. Les classes permettent aussi aux programmeurs d'ajouter d'autres protocoles.

JADE est accompagné de certains outils qui facilitent l'administration de la plate forme et le développement d'applications. Chacun de ces outils est contenu dans un sous package de **jade.tools**.

1. 3. Les agents techniques JADE :

Pour supporter la tâche difficile du débogage des applications Multi-Agents, des outils ont été développés. Chaque outil est empaqueté comme un agent, obéissant aux mêmes règles, aux mêmes possibilités de communication et aux mêmes cycles de vie d'un agent générique. En effet conformément aux standards FIBA, JADE héberge les agents suivants :

1.3.1. Remote Managment Agent (RMA) :

Le RMA permet de contrôler le cycle de vie de la plate forme et tous les agents la composant. L'architecture répartie de JADE permet le contrôle à distance d'une autre plate forme. Un RMA est un objet Java, instance de la classe **jade.tools.rma.rma** et peut être lancé à partir de la ligne de commande : **Java jade.Boot –gui**

Plusieurs RMA peuvent être lancés sur la même plate forme du moment qu'ils ont des noms distincts.

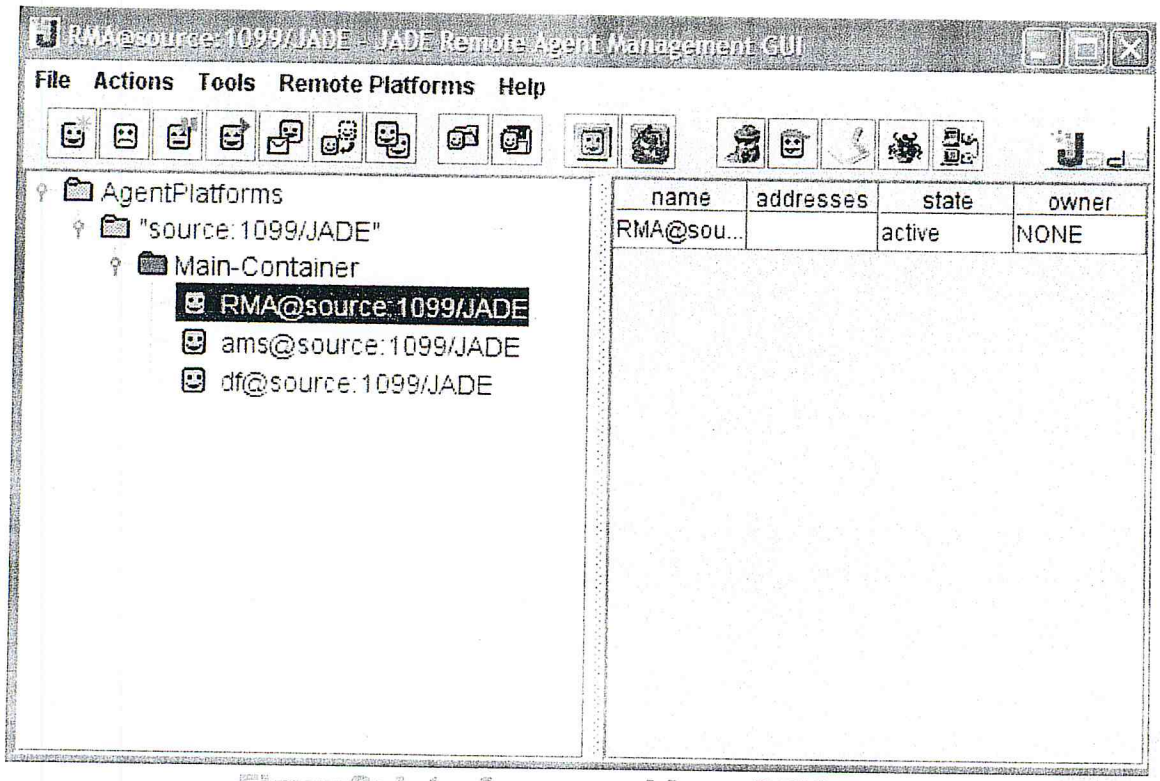


Figure2: Interface graphique RMA

1.3.2. Dummy Agent

L'outil Dummy Agent permet aux utilisateurs d'interagir avec les agents JADE d'une façon particulière. L'interface permet la composition et l'envoi de messages ACL et maintient une liste de messages ACL envoyés et reçus. Cette liste peut être examinée par l'utilisateur et chaque message peut être vu en détail ou même édité. Plus encore, le message peut être sauver sur le disque et renvoyé plus tard. Plusieurs instances du DummyAgent peuvent être lancées de la barre de menu du RMA.

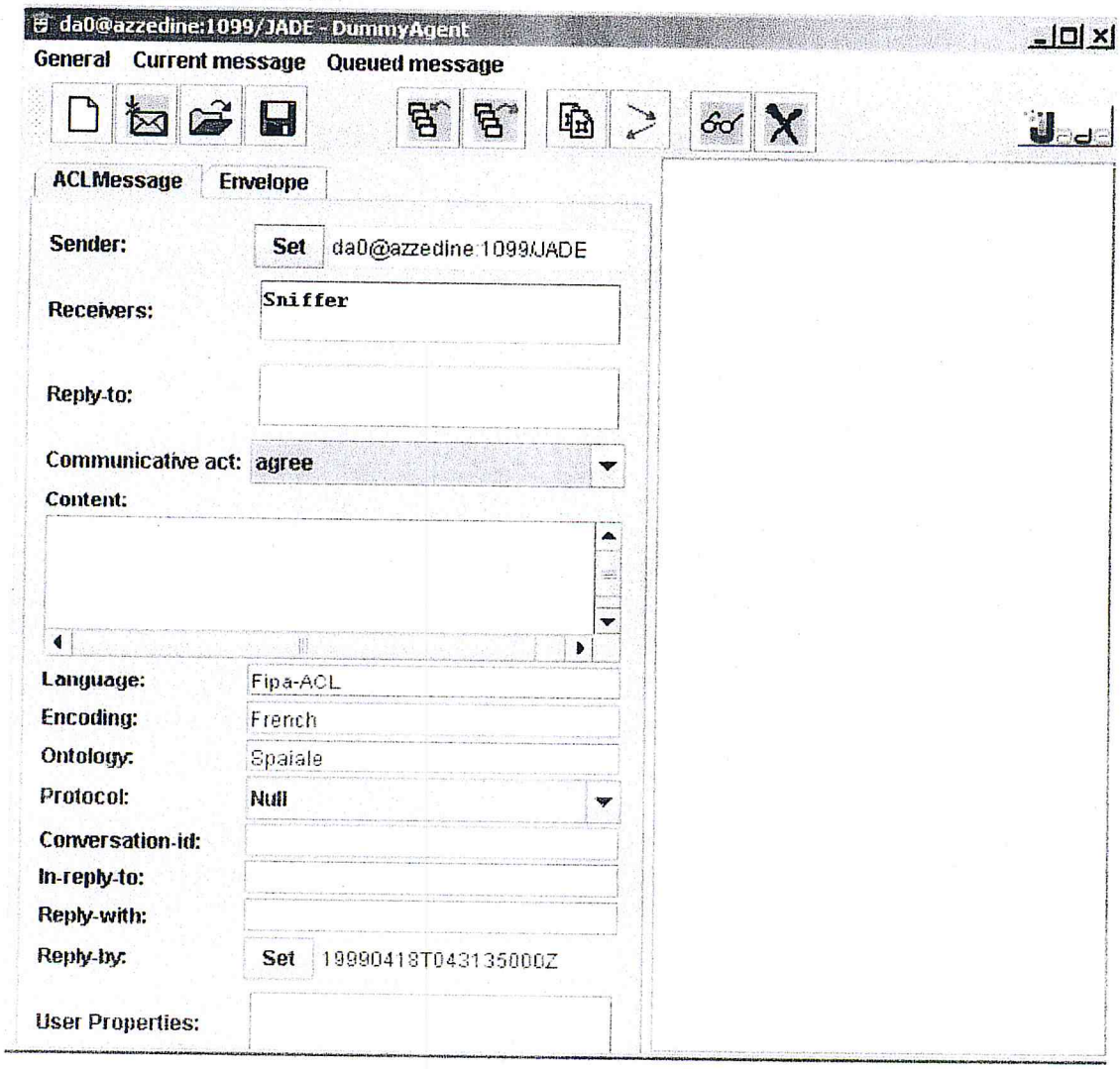


Figure 3 - Interface graphique d'un DummyAgent

1.3.3. Directory Facilitator (DF)

L'interface du DF peut être lancée à partir du menu du RMA .Cette action est en fait implantée par l'envoi d'un message ACL au DF lui demandant de charger son interface graphique. L'interface peut être juste vue sur l'hôte où la plate forme est exécutée. En utilisant cette interface, l'utilisateur peut interagir avec le DF :

- voir la description des agents enregistrés,
- ajouter ou supprimer des agents,
- modifier la description sur les agents enregistrés,
- chercher la description d'un agent.

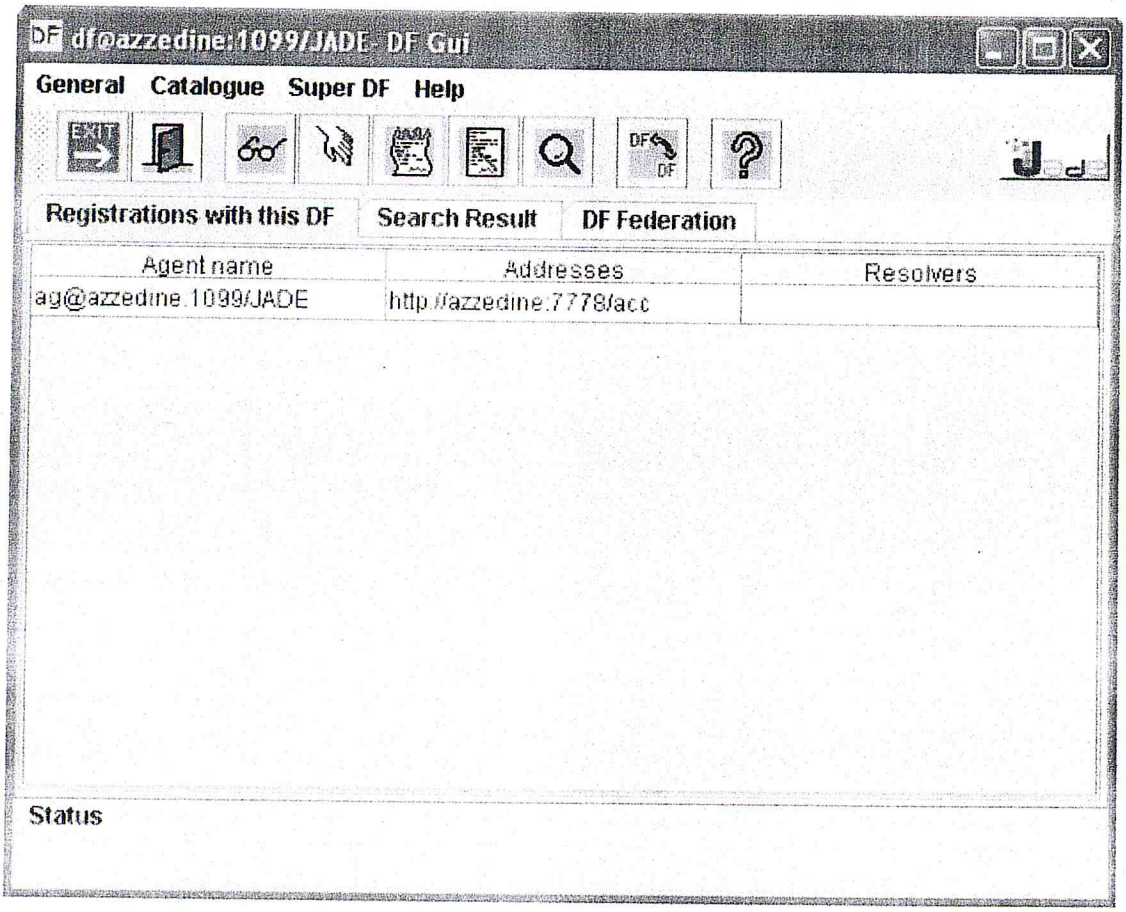


Figure 4 - Interface du Directory Facilitator

1.3.4. Sniffer Agent

Quand un utilisateur décide d'épier un agent ou un groupe d'agents, il utilise un agent sniffer. Chaque message partant ou allant vers ce groupe est capté et affiché sur l'interface du sniffer .L'utilisateur peut voir et enregistrer tous les messages, pour éventuellement les analyser plus tard. L'agent peut être lancé du menu du RMA.

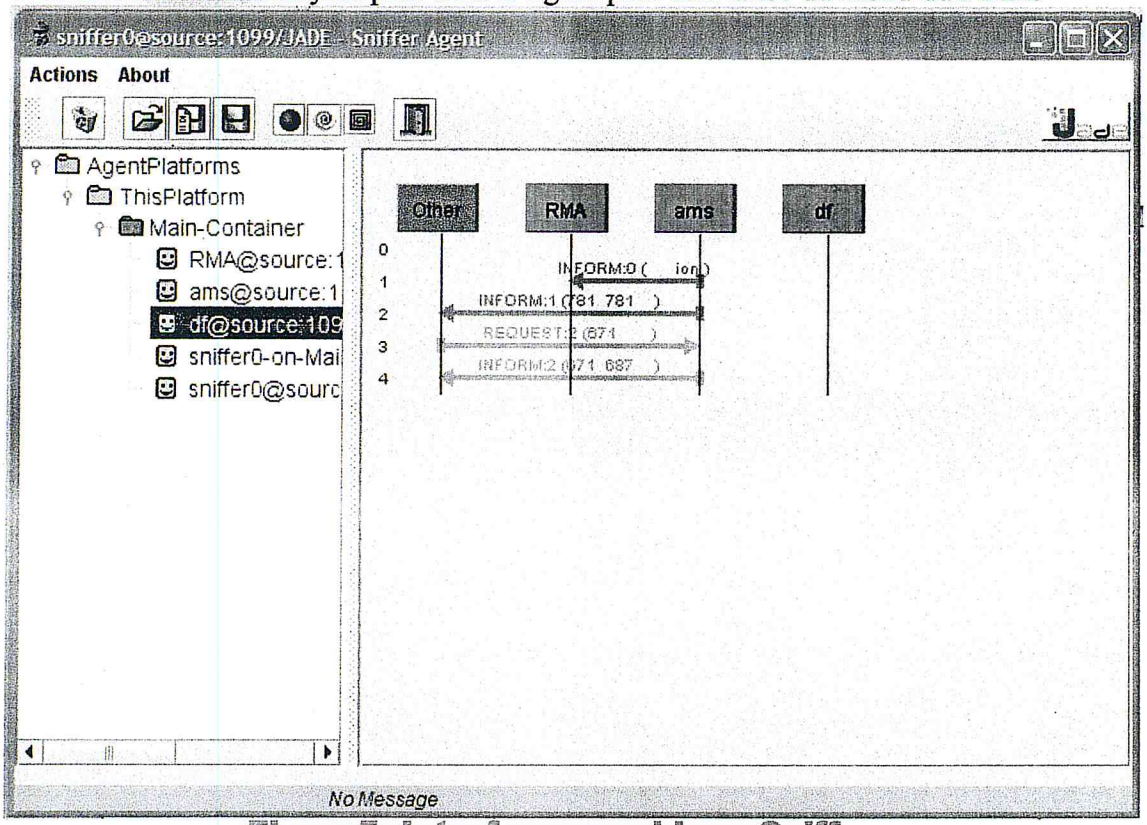


Figure5: Interface graphique Sniffer

1. 3. 5. Introspector Agent

Cet agent permet de gérer et de contrôler le cycle de vie d'un agent s'exécutant et de la file de ses messages envoyés et reçus.

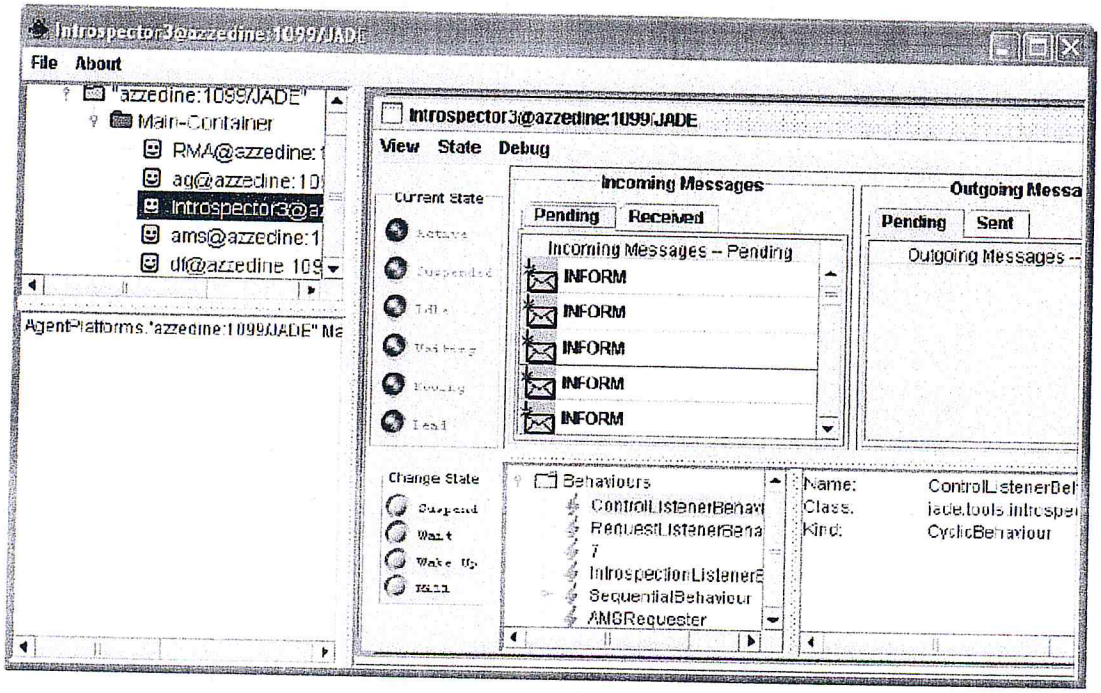


Figure 6 - Interface de l'Introspector

2. Le cycle de vie d'un agent JADE

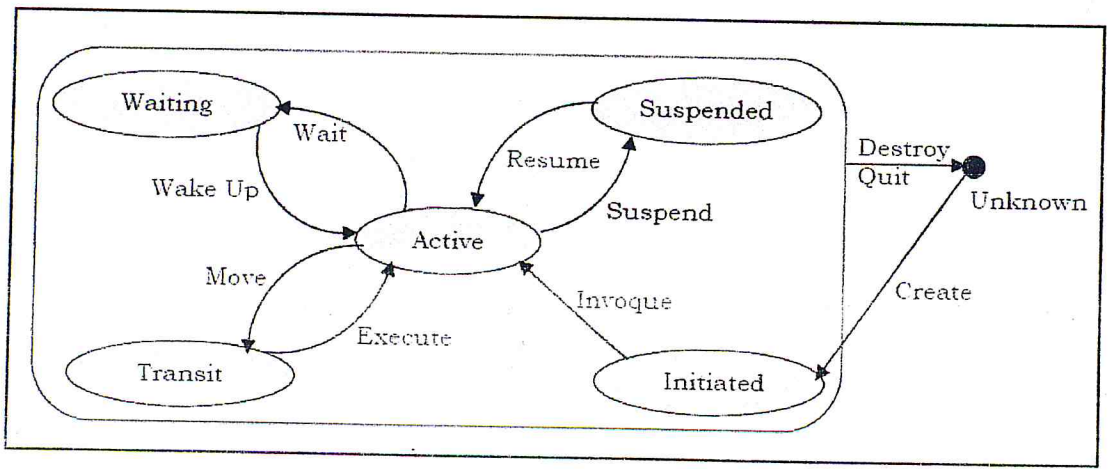


Figure 7 - Le cycle de vie d'un agent

Un agent JADE peut être dans l'un des états suivants :

- **initiated :** L'agent est créé, mais ne s'est pas enregistré encore à l'AMS, il n'a ni un nom ni une adresse et ne peut pas communiquer avec d'autres agents.
- **active :** l'agent est inscrit à l'AMS, a un nom et une adresse réguliers et peut accéder à tous les divers dispositifs de JADE.
- **suspended :** l'agent est actuellement arrêté. Son *thread* est suspendu et aucun comportement d'agent n'est exécuté.
- **waiting :** l'agent est bloqué, attendant quelque chose. Son *thread* est en sommeil et se réveillera quand une certaine condition est vérifiée (par exemple quand un message arrive).
- **deleted :** l'agent est mort. Son thread a terminé son exécution et l'agent n'est plus inscrit à l'AMS.
- **transit:** un agent mobile est dans cet état lorsqu'il migre à un nouvel endroit. Le système continue à stocker ses messages qui seront alors envoyés à son nouvel endroit.

3. Les Behaviours (Comportements

Un comportement représente une tâche qu'un agent peut effectuer. Chaque comportement doit appliquer la méthode **d'action ()**, qui définit les opérations à exécuter par l'agent et la méthode **done ()** qui retourne une valeur booléenne indiquant si un comportement a terminé et doit être enlevé de la file des comportements à effectuer par un agent.

Nous pouvons distinguer trois types de comportements :

- **Behaviours monocoup « One-shot behaviours »** se terminent immédiatement et dont la méthode **action ()** est exécutée seulement une fois. La méthode **done ()** retourne «Vrai».
- **Behaviours cycliques « Cyclic behaviours »** ne sont jamais terminés et dont la méthode **action ()** exécute les mêmes opérations chaque fois qu'elle est appelée. La méthode **done ()** retourne «Faux».

- **Behaviours génériques « Generic behaviours »** exécutent différentes opérations selon un état. Ils se terminent quand une condition donnée est vérifiée.

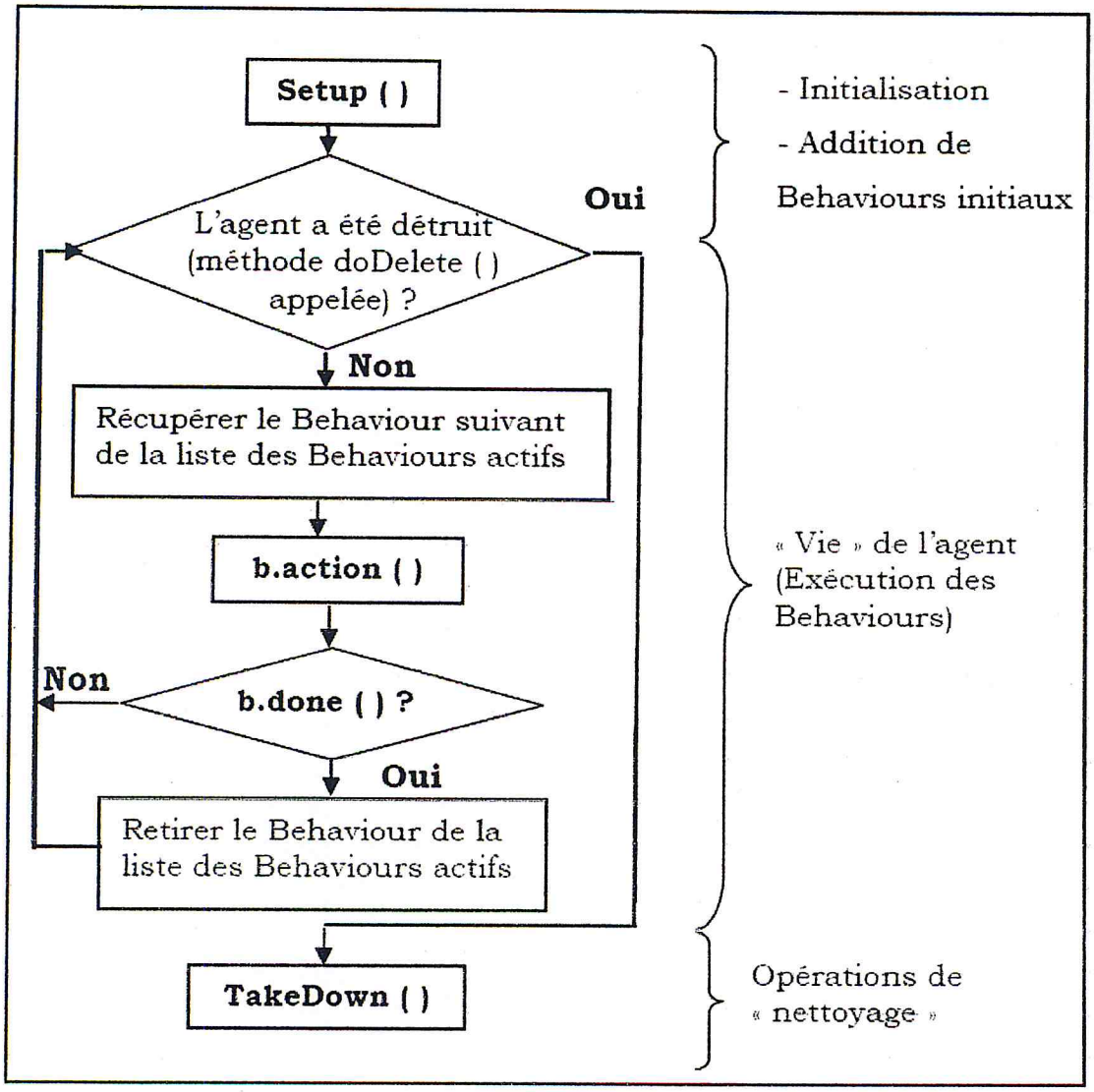


Figure 8 - Exécution d'un *thread* d'agent

4. Communication entre agents

Un des dispositifs les plus importants que les agents JADE fournissent est la capacité de communiquer. Le paradigme de communication adopté est l'envoi asynchrone de message. Chaque agent a une sorte de boîte aux lettres (la file d'attente des messages de l'agent). A chaque fois qu'un message arrive, il est ajouté à la queue de la file et l'agent commence par traiter le premier message arrivé.

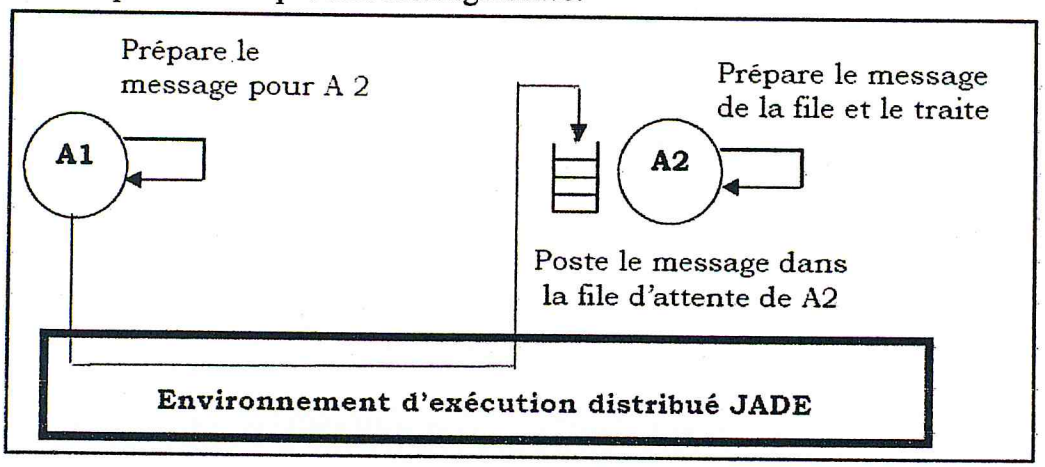


Figure 9 - Le paradigme JADE de traitement asynchrone de messages.

Les messages échangés par des agents JADE ont un format indiqué par la langue FIPA-ACL *(cf.Partie I .Chapitre 4. § VI. 2).*

4. 1. Envoi d'un message

L'envoi d'un message consiste à créer un objet **ACLMessage** et appeler la méthode **send ()** de classe Agent.

Exemple : Le code suivant correspond à l'envoi d'un message à l'agent « Peter » dans le but de l'informer qu' « Aujourd'hui il pleut ».

```
ACLMessage msg=New ACLMessage (ACLMessage.INFORM);
msg.addReceiver (new ("IMENE"),AID.ISLOCALNAME));
msg.addReceiver (new ("IMENEK"),AID.ISLOCALNAME));
msg.SetLanguage("Francais") ;
msg.SetOnthology("Météo" ) ;
msg.SetContent("Aujourd'hui il pleut");
```

4. 2. Réception d'un message

La lecture des messages de la file d'attente de message est accomplie par la méthode **receive ()** de la classe d'agent.

```
ACLMessage msg=receive ();
      if (msg!=null){
      //traitement de message
      }
```

4. 3. Blocage d'un comportement attendant un message

Le Behaviour qui traite les messages entrants ne sait pas exactement quand un message arrivera. Il devrait voter la file d'attente de message en appelant sans interruption **myAgent.receive()**. Ceci naturellement gaspillerait complètement le TEMPS- CPU. La méthode **block ()** de la classe Behaviours enlève un Behaviour de la file d'agents et le met dans un état bloqué. Chaque fois qu'un message est reçu tous les comportements bloqués sont insérés en fin de file d'agents et ont une chance de lire et traiter le message.

```
public void Action () {
ACLMessage msg=myAgent.receive();
        if (msg!=null){
        //traitement de message

        }
                else{
                        block ();
                }
}
```

C'est le modèle vivement recommandé pour recevoir des messages dans un comportement

4. 4. Lecture sélective de la file d'attente des messages

La méthode **receive ()** renvoie le premier message dans la file d'attente de messages et le supprime de la file. S'il y a deux ou plus de Behaviours recevant des messages, un des Behaviours peut " voler " un message qui intéressait un autre. Pour éviter ceci, il est possible de lire seulement des messages avec certaines caractéristiques (par exemple en spécifiant l'expéditeur) indiquant un paramètre de jade.lang.acl.MessageTemplate dans la méthode de receive ().

C'est le modèle vivement recommandé pour recevoir des messages dans un comportement

```
public void Action () {

MessageTemplate template =MessageTemplate.MatchProtocol("RechercheDeProduit");
ACLMessage msg=myAgent.receive(template);

if (msg!=null){
//traitement de message
}
        else{
                block ();
        }
}
```

Conclusion :

JADE est une plateforme Open Source et été utilisé dans plusieurs projets, de plus plusieurs référence bibliographique et des tutoriels peuvent être consultés afin de mieux cerner le fonctionnement de JADE.

Annexe 2 :

AUML

Introduction :

Les faiblesses d'UML pour la représentation des systèmes multi-agents ont conduit une équipe de chercheurs travaillant dans différentes entreprises ou universités (Siemens, University Paderborn, Intelligent Automation, Fujitsu…) à concevoir AUML.

L'objectif est de mettre au point des sémantiques communes, des méta-modèles et une syntaxe générique pour les méthodologies agents. AUML est un des fruits de la coopération entre FIPA (Foundation of Intelligent Physical Agents) et l'OMG (Object Management Group). Par rapport à UML, Agent-UML propose des extensions pour la représentation des agents que nous détaillerons dans cette annexe.

Principes sur UML

UML (Unified Modeling Language) **[Ode98],** unifie et formalise les méthodes de plusieurs approches orientées objets. UML prend en charge plusieurs types de modèles :

- **les cas d'utilisation** : la spécification des actions que le système ou la classe peut réaliser en interaction avec les acteurs extérieurs. Ils sont souvent utilisés pour décrire comment un utilisateur communique avec son logiciel.
- **les modèles statiques** : ils décrivent la sémantique statique des données et des messages d'une manière conceptuelle et d'une manière plus proche de l'implémentation (il s'agit des diagrammes de classes et de packages).
- **les modèles dynamiques** : ils incluent les diagrammes d'interaction (i.e., les diagrammes de séquence et les diagrammes de collaboration), les schémas d'état et les diagrammes d'activité.
- **les modèles d'implémentation** : ils décrivent la répartition des composants sur différentes plateformes (i.e., les modèles de composants et les diagrammes de déploiement).

2. Déficiences d'UML :

L'idée générale d'AUML est de combler les déficiences d'UML pour la modélisation des systèmes à agents. Parmi ces déficiences, on trouve **[Mul97]** :

- Des relations entre classes statiques (agrégation, généralisation, et association) mais qui semblent tout de même adéquats. Il est possible d'utiliser des associations de classes et des stéréotypes pour étendre UML avec des relations spécifiques pour les agents.

- Les accointances sont des relations importantes entre agents. Il s'agit d'une relation dynamique entre des instances et UML n'est pas très adapté pour les représenter.

- Un certain nombre de concepts de haut niveau (comme les engagements, les contrats, etc.) peuvent être relativement bien représentés avec UML mais d'autres (comme les croyances et les intentions) ne le peuvent pas.

- Il est difficile de représenter l'état interne des agents. Il faudrait un modèle proposant des concepts de haut niveau « cognitif » (BDI, BC, GAP, …).
- UML n'est pas efficace pour représenter des connaissances fonctionnelles (buts, planification, process, etc.). Pourtant beaucoup de méthodologies agents utilisent les buts et la décomposition de buts en sous-buts.

- Il n'est pas évident que les approches de modélisation à états finis soient adaptées pour les agents. Les agents ont des espaces d'états vastes qu'il n'est pas évident de partitionner en un plus petit nombre de macro-états de plus haut niveau. Les agents peuvent apprendre et s'adapter à différentes choses et des paramètres comme les croyances interagissent pour influencer le comportement de façons subtiles. Ces systèmes sont dynamiques, non linéaires et ont un comportement émergeant.

3. AUML :

AUML **[Ode00]** est basé sur la méthode UML (Unified Modeling Language) qui est une méthode de génie logiciel utilisée pour les développements en langages orientés-objets. Elle est déjà largement utilisée par la communauté des concepteurs-objet et son succès continue de croître.

Comme nous l'avons déjà vu, par rapport aux objets, les agents ont des activités autonomes et des buts. C'est cette différence qui entraîne l'insuffisance d'UML pour modéliser les agents et les systèmes multi-agents. Aussi AUML remplace-t-il la notion de méthode par celle de service. Ses principales extensions sont :

- Diagramme de classes d'agent qui est une reformulation du diagramme de classes d'objets,
- Diagramme de séquence qui permet une meilleure modélisation des interactions entre agents,
- Diagramme de collaboration qui complémente le diagramme de séquences en proposant une autre lecture et vision des interactions entre agents.

4.1. Le diagramme de classes d'agent :

Un diagramme de classes agent doit supporter tous les concepts liés à l'agent, ce diagramme est illustré dans la figure suivante :

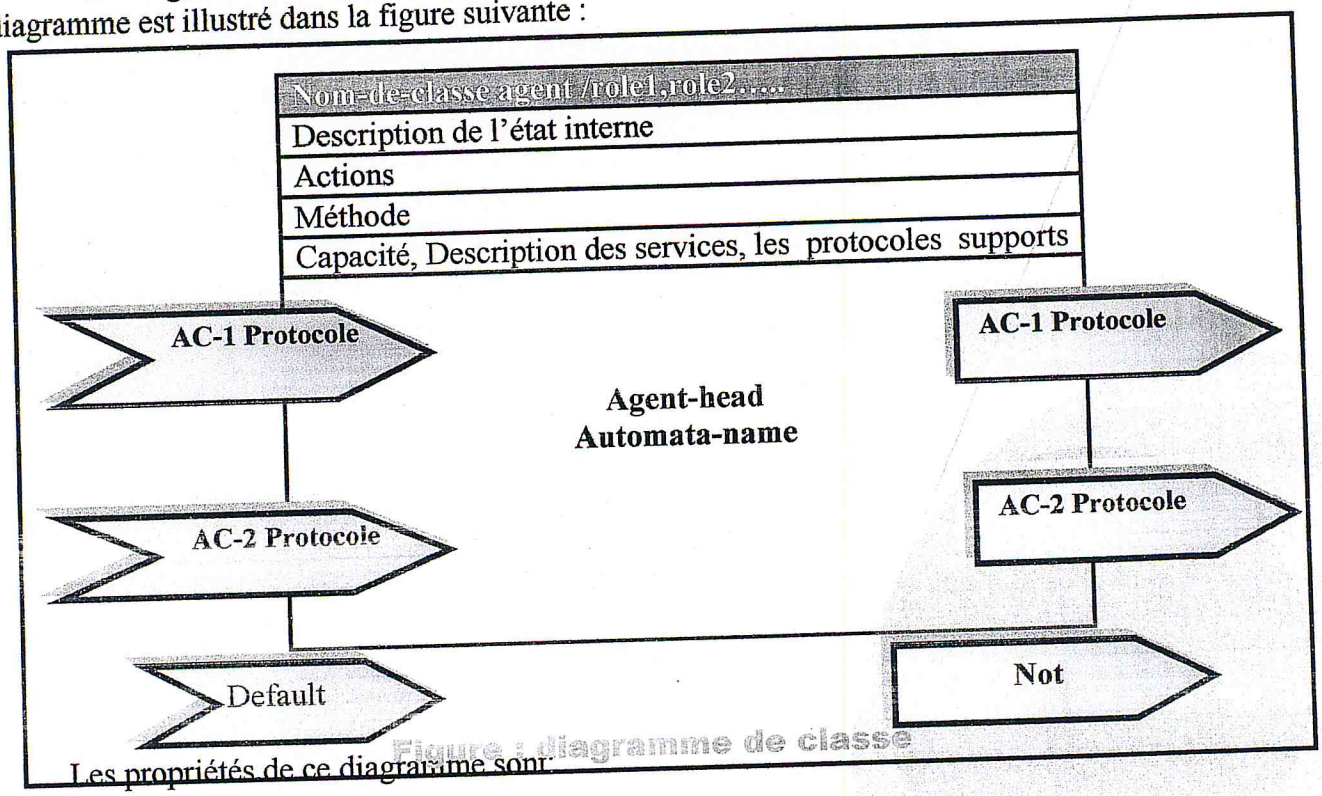
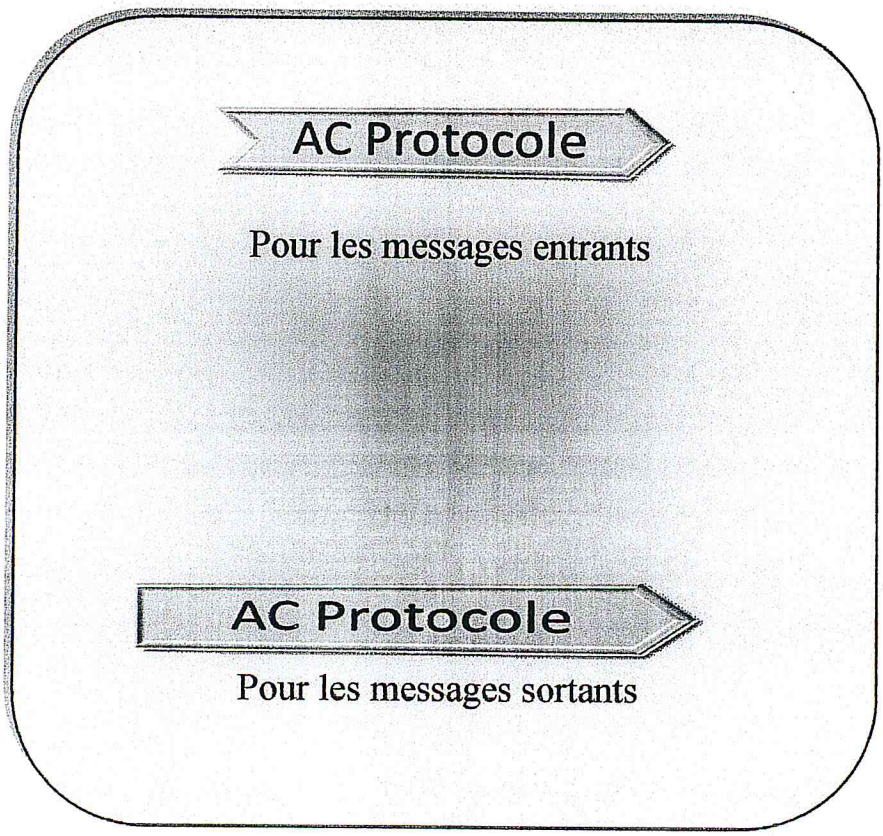


Figure : diagramme de classe

Les propriétés de ce diagramme sont:

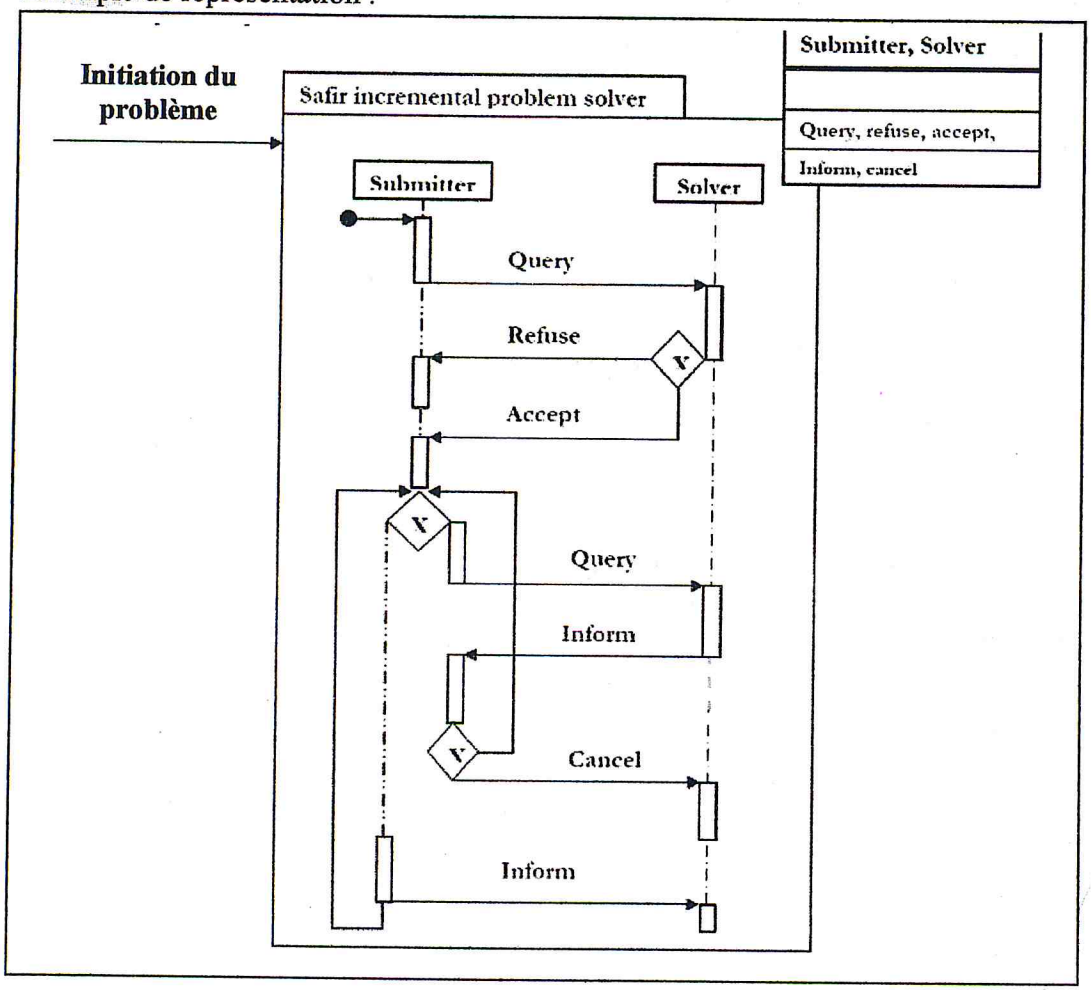
- **Nom de la classe agent/ rôle1, rôle2,...:** Un agent d'une classe donnée peut avoir plusieurs rôles (un détaillant peut être acheteur ou vendeur) .
- **Description des états:** Définition de variables d'instance qui reflètent l'état de l'agent,
- **Actions(Plans):** deux types d'actions peuvent être spécifiés: action **pro-active** exécutée par l'agent lui même si une précondition devient vraie, et **ré-active** résultant d'un message reçu d'un autre agent. En d'autres termes les actions sont les plans qu'a un agent.
- **Méthodes:** Elles sont définies comme dans UML, avec éventuellement des pré-conditions, post-conditions ou invariants.
- **Envoi et réception de messages:** La principale interface entre les agents et leur environnement est la réception et l'envoi de messages, ces messages peuvent être des classes ou des objets. Les messages sont représentés par :



- **Agent-head-automata:** représente les changements d'état induits par les échanges de messages. Il relie les messages entrants avec l'état interne, actions, méthodes et les messages sortants.

4.2. Protocoles d'interaction (AIP : Agent Interaction Protocol):

Exemple de représentation :



Dans le protocole de la figure ci-dessus, aucune précision n'est donnée sur le traitement ou la construction des messages :

- La construction de la requête (par Submitter) peut être un processus complexe décrit par un diagramme d'activités ;
- Le traitement de cette requête (par Solver) peut être décrit par un autre diagramme d'activités ou de séquence.

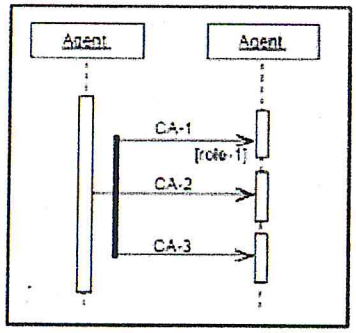
Le schéma ci-dessus est constitué de trois couches. Ce découpage en couches réifie les **processus inter-agents** et **ceux internes** à chaque agent :

1- le protocole est la couche supérieure représentée sous forme d'un package ou d'un Template,

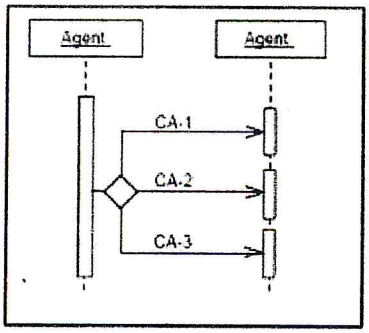
2- Les échanges entre agents sont la deuxième couche :

4.3. Diagramme de séquences :

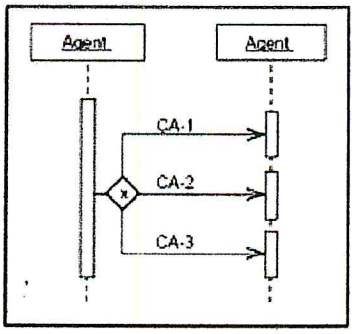
Nommage des acteurs : nom agent/rôle : Class
Les messages échangés ne sont plus des noms de méthodes mais des actes de Communication, Utilisation des **threads d'interactions** concurrents : AND, OR, XOR



AND : Tous les messages sont envoyés à la fois



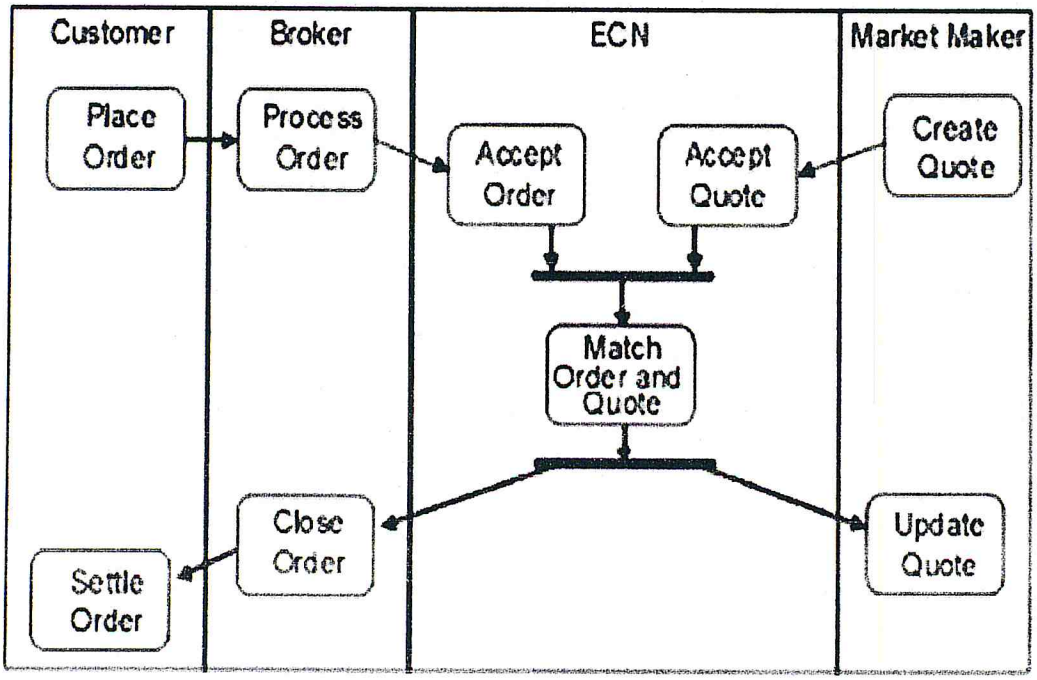
OR : Un ou plusieurs messages sont envoyés



XOR : Un seul message est envoyé

4.4. Digramme d'activité :

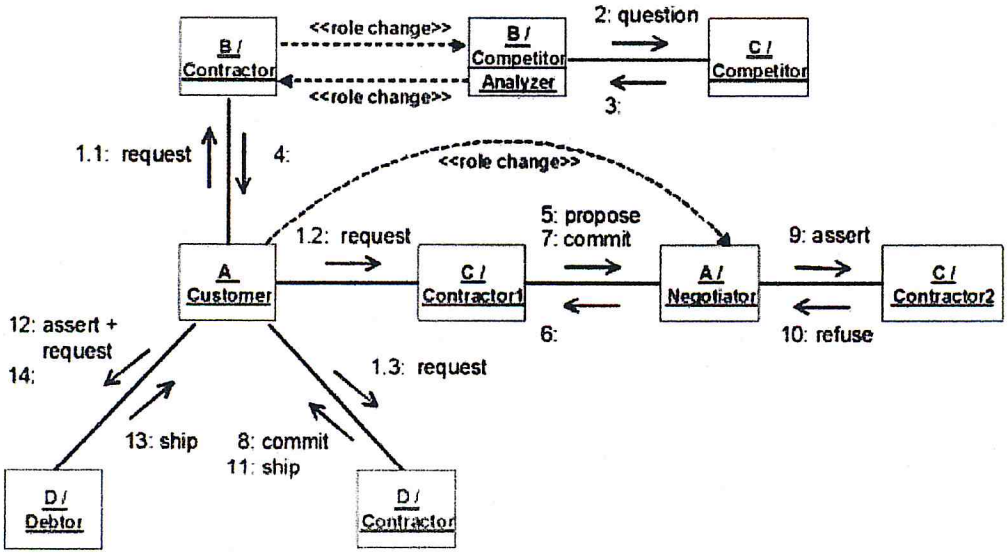
Décrit les opérations entre agents et les événements qui le déclenchent.



ECN: Electronic Commerce Network.

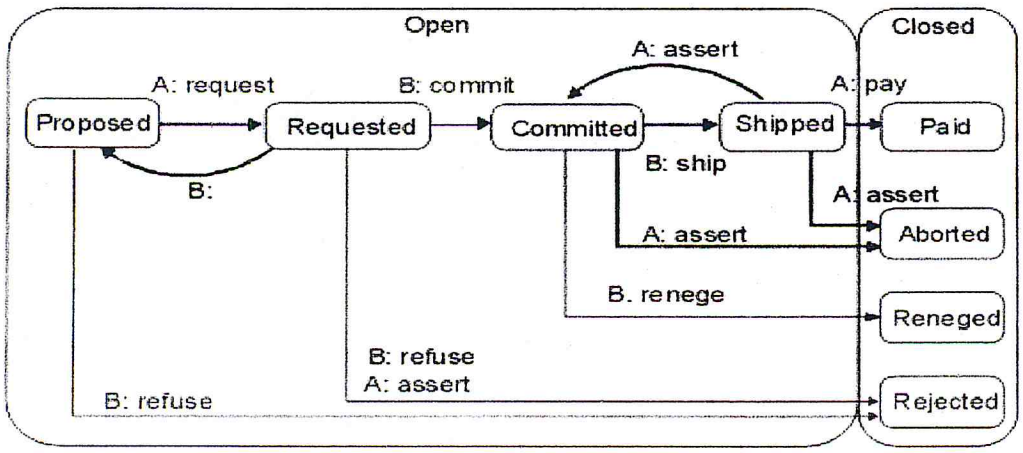
4.5. Diagramme de collaboration:

Présente les interactions entre les objets :



On remarque que les agents sont placés n'importe où dans le diagramme, les messages échangés sont numérotés et par rapport au diagramme de séquence, les protocoles d'interactions sont moins clairs.

4.6. Diagramme d'états- transitions : Décrit les changements d'état lors des échanges entre agents



Les traitements internes de l'agent constituent la troisième couche :

- Diagramme d'activité pour chaque agent,
- Diagramme d'état transition pour chaque agent.

5. Limitations d'AUML :

Selon **[Par03]**, AUML présente les limitations ci-après:

- Les diagrammes sont désordonnés et peuvent être mal interprétés ;
- L'expression de toutes les informations nécessaires sur les protocoles peut les rendre illisibles;
- Les cas de redondance sont difficiles à identifier et corriger ;
- La description des actions temporelles (telles que le timeout, deadline, ...) est difficile à exprimer
- La notion d'historique des échanges n'existe pas ;
- La terminaison des interactions n'est pas toujours spécifiée, surtout

Conclusion :

Aujourd'hui AUML n'est pas encore un langage de modélisation fini et adopté par la communauté car aucune spécification finale n'a été publiée officiellement mais plusieurs publications sur des extensions d'UML ont étés publiées par les membres du groupe.

Actuellement, leurs travaux se concentrent sur les interactions, plus spécifiquement sur les protocoles d'interaction, mais aussi sur d'autres aspects connexes comme les langages d'interaction, la coordination, les rôles des agents. D'autres travaux portent également sur les architectures, et les ontologies.

Bibliographies.

[STE00] Stéphane Vialle

2A-SI - Réseaux : Programmation par RPC et JavaRMI

http://www.metz.supelec.fr/~vialle

[LIO04] Lionel Seinturier

Architecture CORBA 02/03/04

Université Pierre & Marie Curie

Lionel.Seinturier@lip6.fr

BEL04] Fabio Bellifemine, Giovanni Caire, Dominic Greenwood **Juin 2004**

Devellopping multi-agent technologie with JADE,

Wiley Series in Agent Technology

Series Editor: Michael Wooldridge, *Liverpool University, UK*

[PTP06] DIJOUX Alexandre 10608162 **12/12/2006**

EMMA Samuel 10608165

http://www.openp2p.com

[GEO01] George et Olivier GARDARIN

Le CLIENT-SERVEUR

ISBN : 2 212 088760.

[PIE02] Pierre-Yves Cloux, David Doussot

Technologies et Architectures Internet 2 °ème édition

ISBN 2 20 008178 0

[Bou04] M. Boughanem,UniversitéPaul Sabatier de Toulouse 2004
Laboratoire IRIT
bougha@irit.fr

[Jas00] JASIST : Journal of the American Society for Information Scienceand Technology

[Bae92] Frakesand Baeza-Yates, eds. Information Retrieval: Data Structures & Algorithms, *PrenticeHall* *1992*

[Cal02] A Multi-Agent Personal Information Retrieval System Travis Bauer and David B. Leake Indiana University 2002

[Tro06] Une architecture multi-agent pour la recherche sur Internet Marc Côté, Nader Troudi

2006

[Jar01] Systèmes multi-agents : Principes généraux et applications B. Chaib-draa, I. Jarras et B. Moulin

2001

[Fis79] Http://www.engin.umd.umich.edu/CIS/course.des/cis479/projects/FISA.html

[Feg01] Web Economics: A Case for Agent-based Digital Libraries I. Ferguson, J. P. Mueller, M. Pischel, and M. Wooldridge.

2001

[Pic03] David Picard, M. Cord, and Arnaud Revel.

Cbir in distributed databases using a multi-agent system.

2003

[Ode98] J. Odell and M. Fowler. *Advanced object-oriented analysis and design using UML*. SIGS Books / Cambridge University Press

1998

[Ode00] J. Odell, H. v. D. Paranuk, B. Bauer: *Representing Agent Interaction Protocols in UML*,submitted for Autonomous Agents,2000

2000

[Par03] H. V. D. Parunak, and J. Odell. *Engineering Artifacts for Multi-Agent Systems*, ERIM CEC

2003

[Mül97] J. P. Müller. The Design of Autonomous Agents : A Layered Approach, volume 1177 of Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg,

1997