Université Constantine 2

Faculté des NTIC

Département d'Informatique Fondamentale et ses Applications

1ère Année Master Réseaux et Systèmes Distribués
TP Algorithmes Distribués (ALDI)

TP1 : Initiation à la plateforme JADE (Création et exécution des agents)

Compétences à acquérir

Suite à ce TP, vous serez en mesure de :

- Se familiariser avec la plateforme JADE.
- Comprendre le principe de fonctionnement d'un système multi agents.
- Installer la plateforme JADE.
- Créer un agent et lancer son exécution pour afficher son nom local et son identifiant.
- Créer un agent qui reçoit des paramètres passés en arguments puis les afficher.

Introduction

- ✓ JADE (Java Agent Development Framework) est un environnement de développement d'agent implémenté totalement dans le langage JAVA.
- ✓ Il facilite la mise en place d'un système multi-agent (SMA) répondant aux spécifications FIPA (Foundation for Intelligent Physical Agents) à travers un ensemble d'outils.
- ✓ La plateforme JADE inclut des composants qui contrôle un système multi-agent :

  Un runtime Environment : l'environnement où les agents peuvent vivre, il doit être activé pour pouvoir lancer les agents.

  Une librairie de classes : que les développeurs utilisent pour écrire leurs programmes.

  Une suite d'outils graphiques : qui facilitent la gestion et la supervision de la plateforme des agents.

Introduction

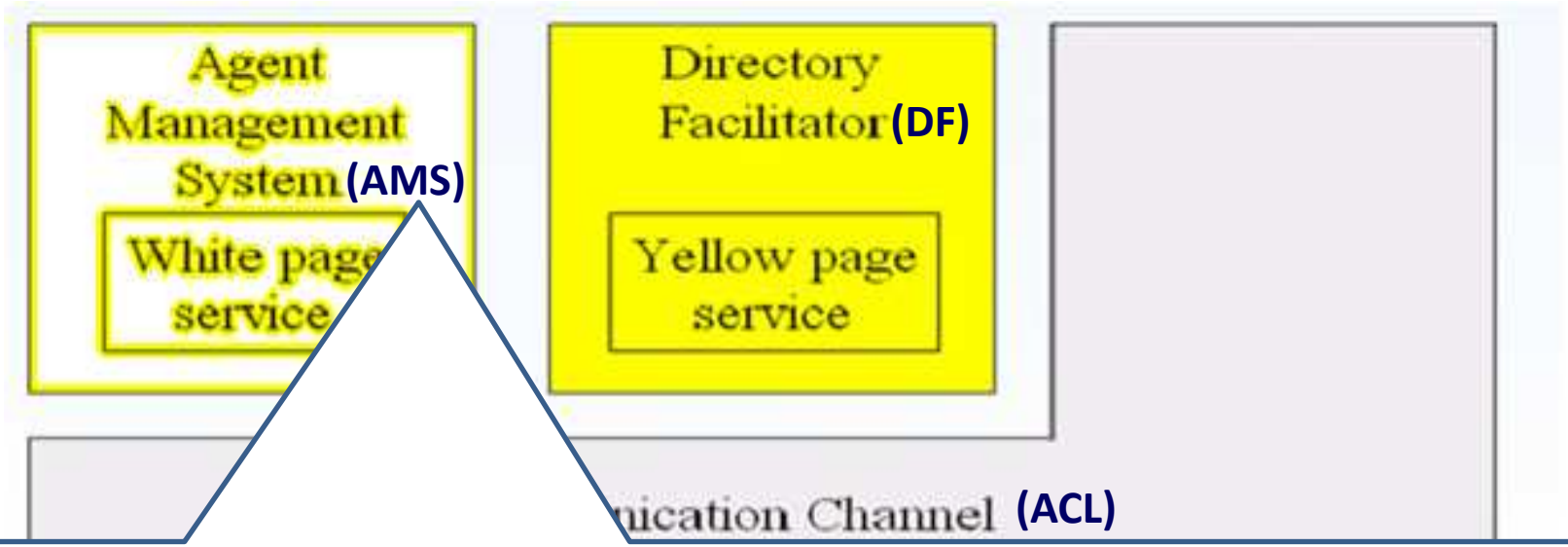
- ✓ Chaque instance du JADE est appelée conteneur « container », elle peut contenir plusieurs agents.
- ✓ Un ensemble de conteneurs constituent une plateforme.
- ✓ Chaque plateforme doit contenir un conteneur spécial appelé « main-container » et tous les autres conteneurs s'enregistrent auprès de celui-là dés leur lancement.
- ✓ Un « main-container » contient trois agents spéciaux appelés **AMS, DF** et **ACL** qui sont créés automatiquement au lancement du « main-container »
  - **AMS** (Agent Management System)
  - **DF** (Directory Facilitator)
  - **ACL** (Agent Communication Channel)

Introduction

Architecture du main-container

Introduction

Architecture du main-container



- **AMS** (Agent Management System) permet de :

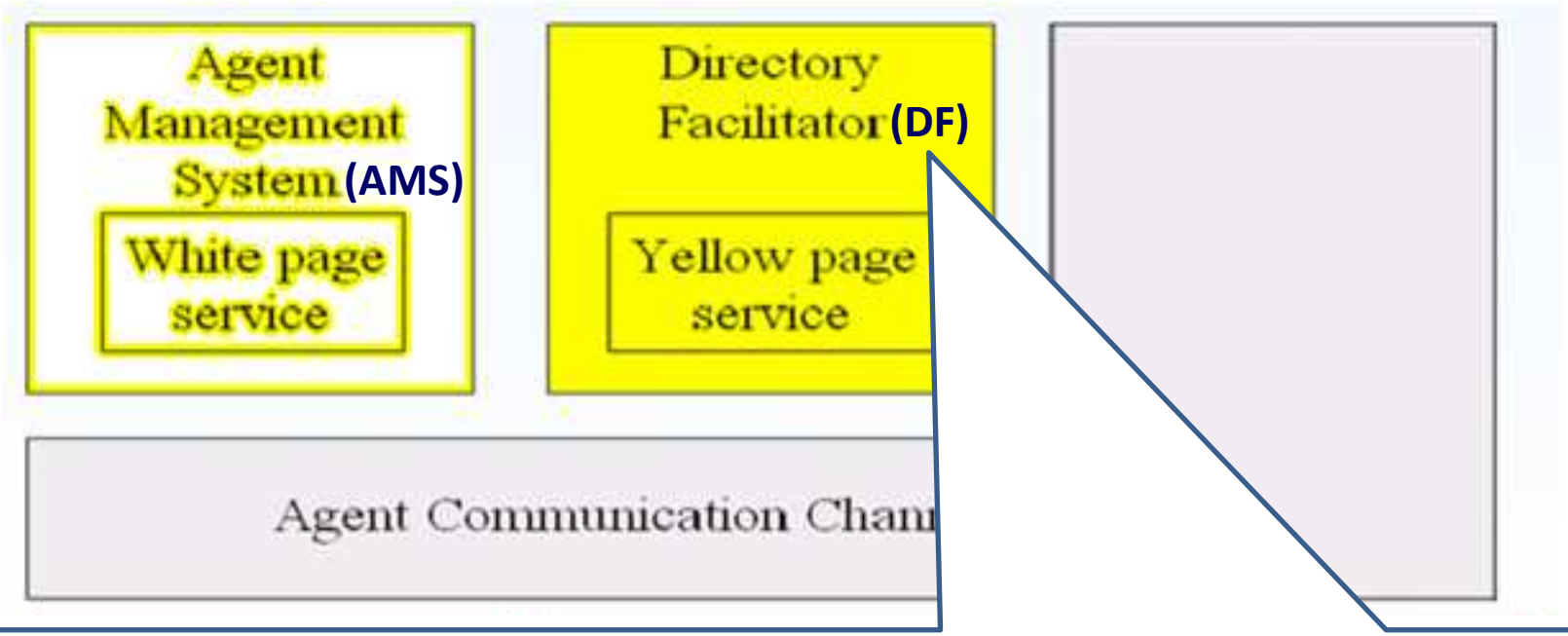
✓Gérer le cycle de vie des agents

✓Maintenir une liste de tous les agents qui résident sur la plate-forme (pages blanches)

✓Contrôler l'accès ainsi que l'utilisation du canal de communication des agents

Introduction

Architecture du main-container

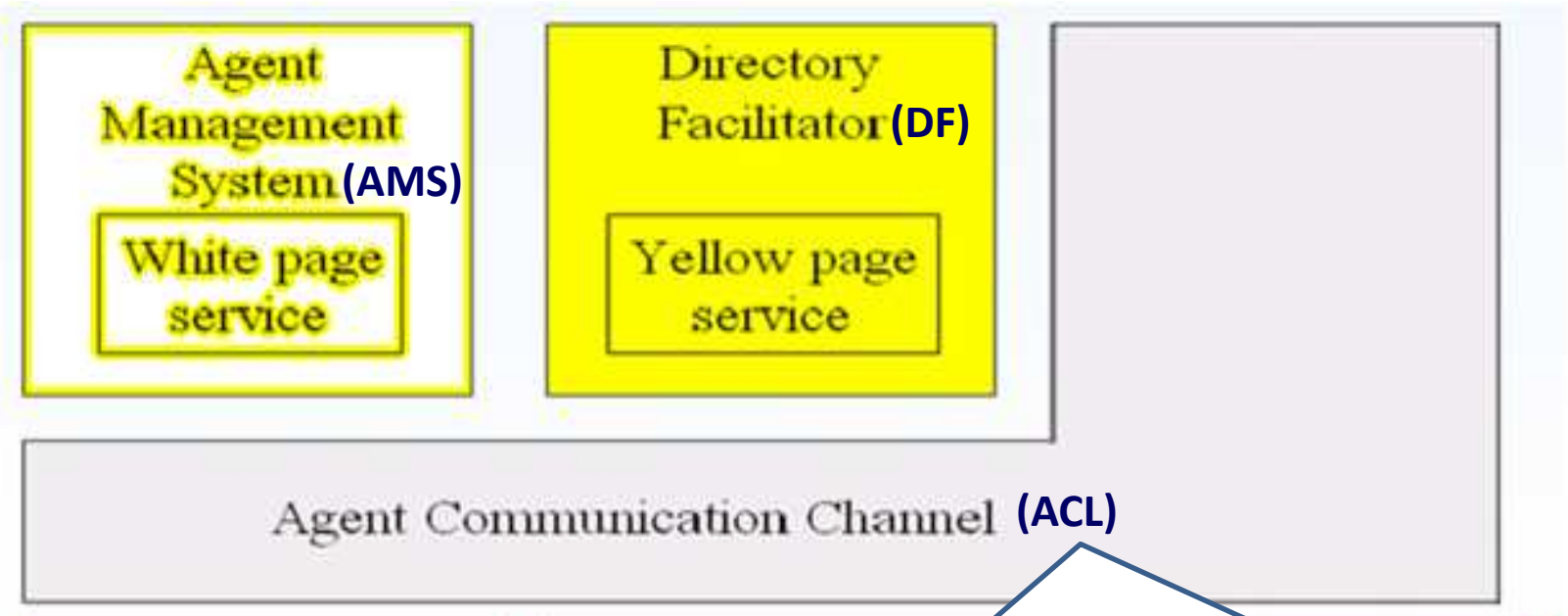


DF (Directory Facilitator) permet de :

- ✓Enregistrer les services offerts par les agents (Pages jaunes)
- ✓Rechercher les services offerts par les agents

Introduction

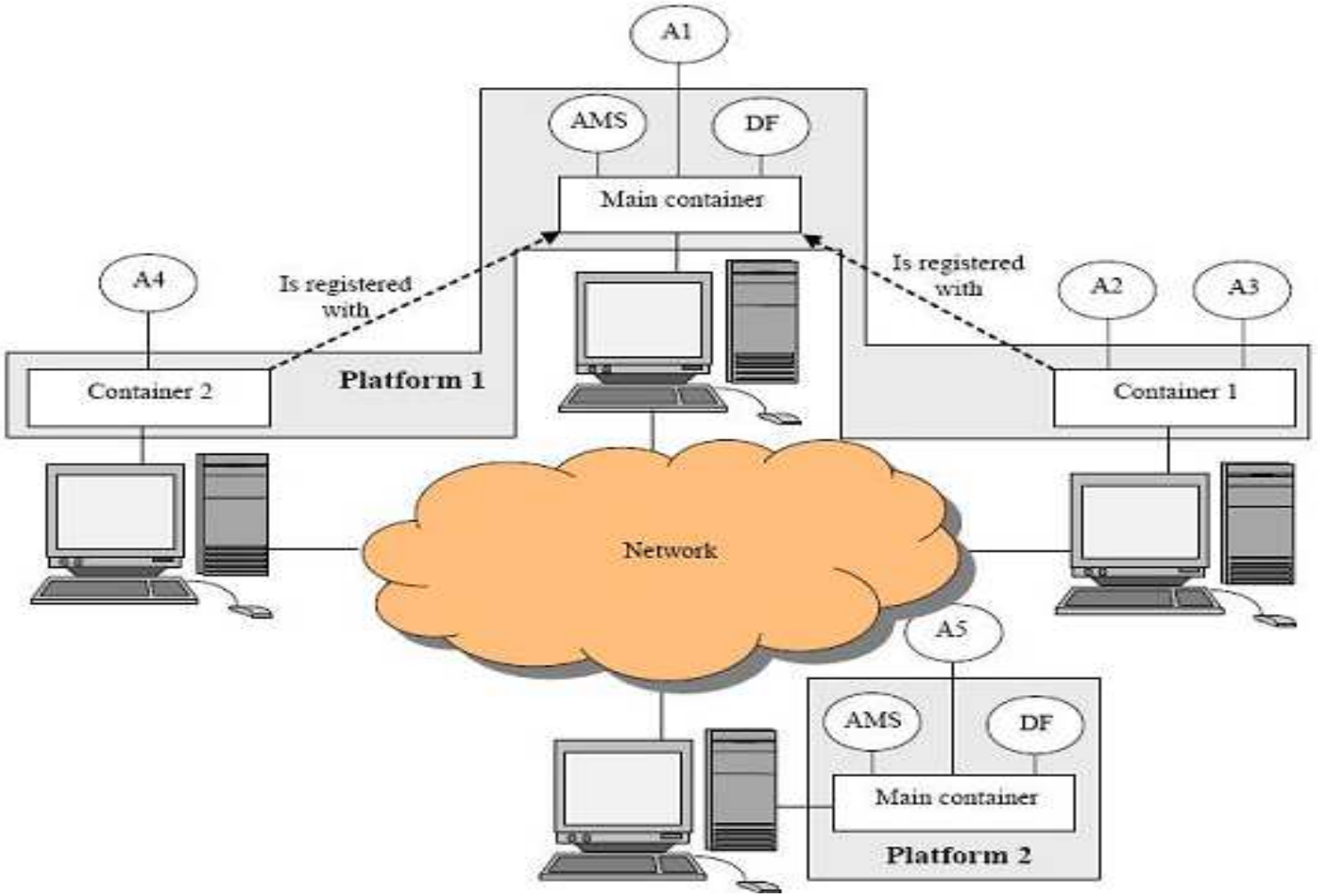
Architecture du main-container



- **ACL** (Agent Communication Channel) permet de :
  - ✓Gèrer les communications entre les agents

Introduction

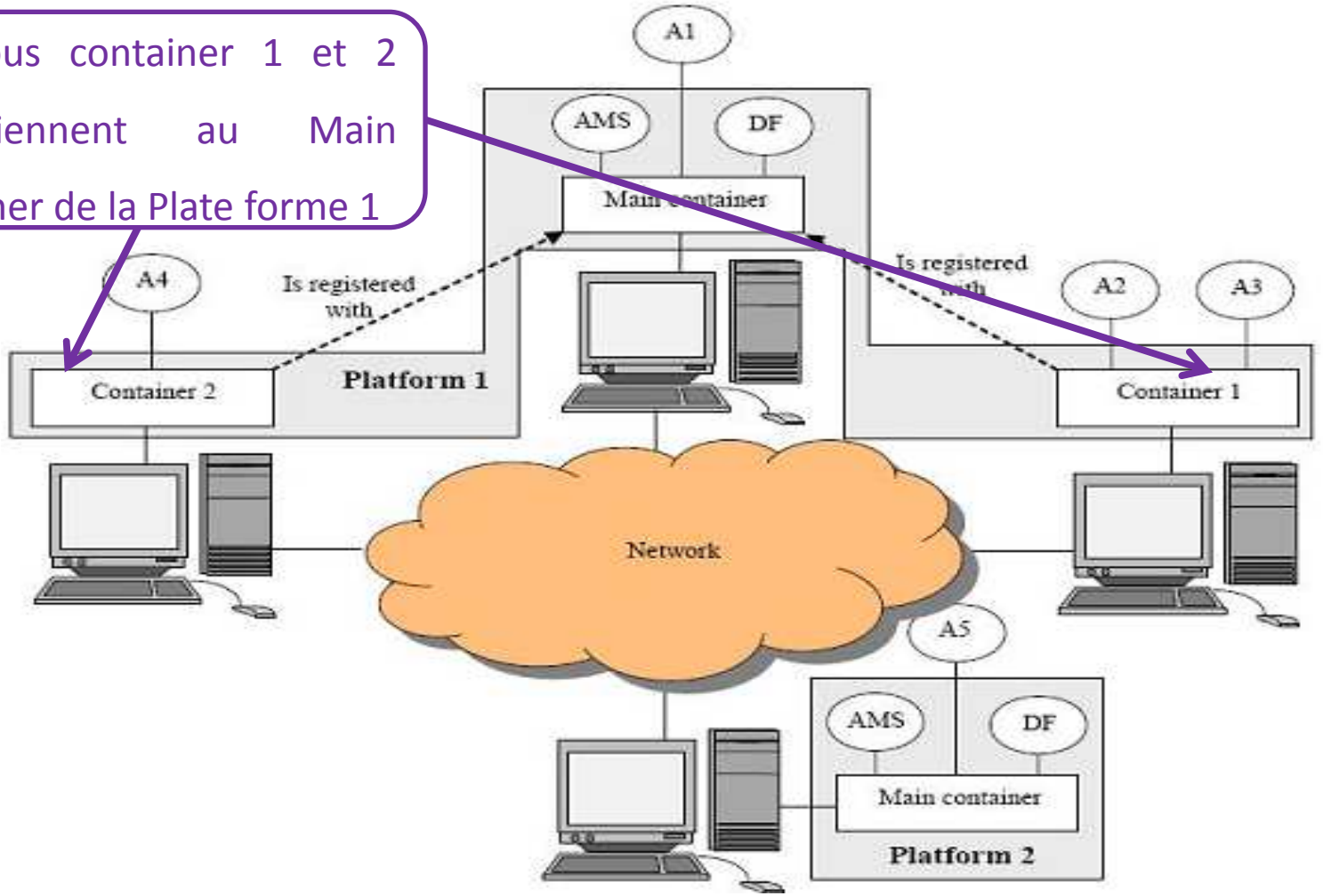
Exemple de Containers (Environnement d'exécution pour les agents)



Introduction

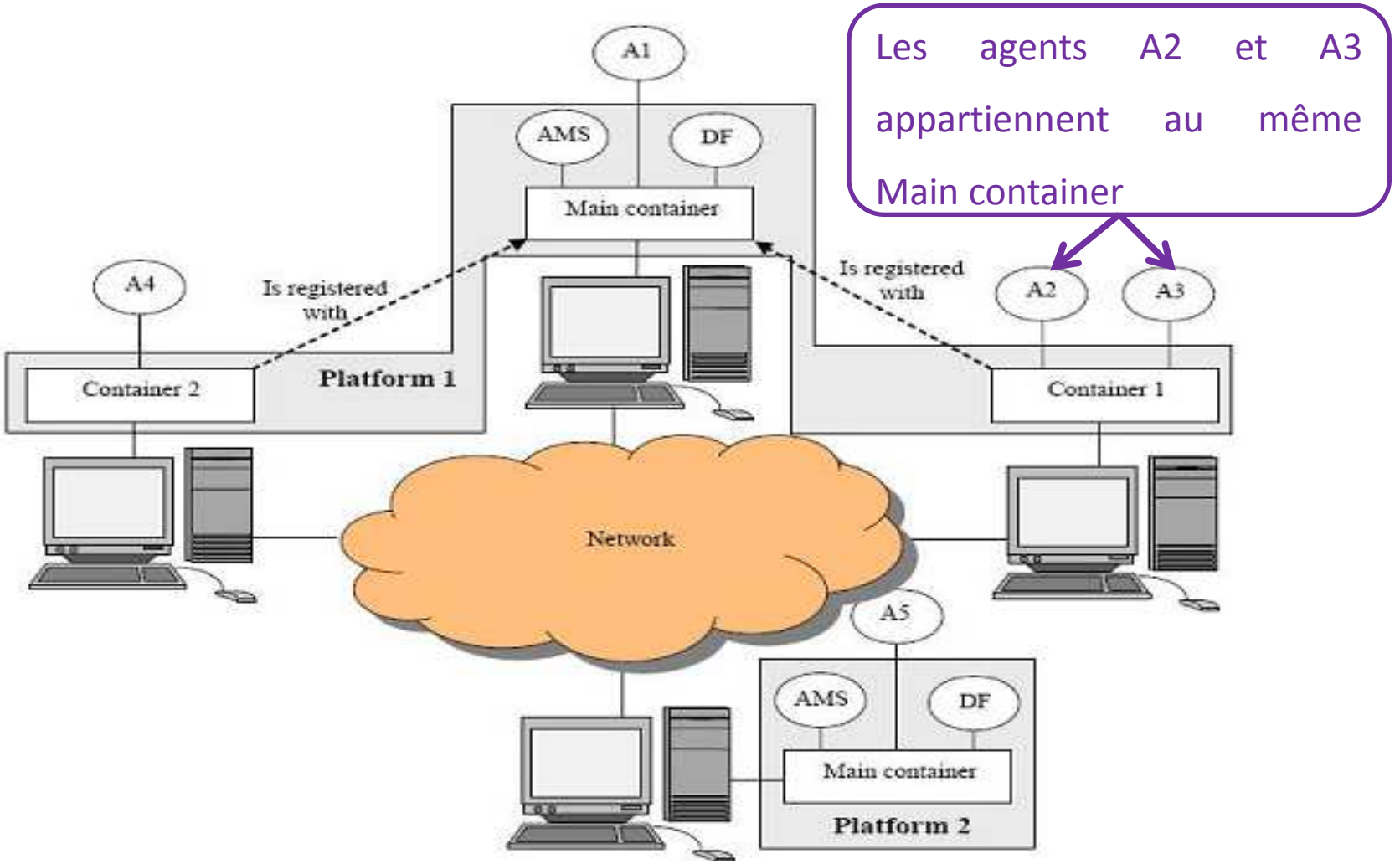
Exemple de Containers (Environnement d'exécution pour les agents)

Les sous container 1 et 2 appartiennent au Main container de la Plate forme 1



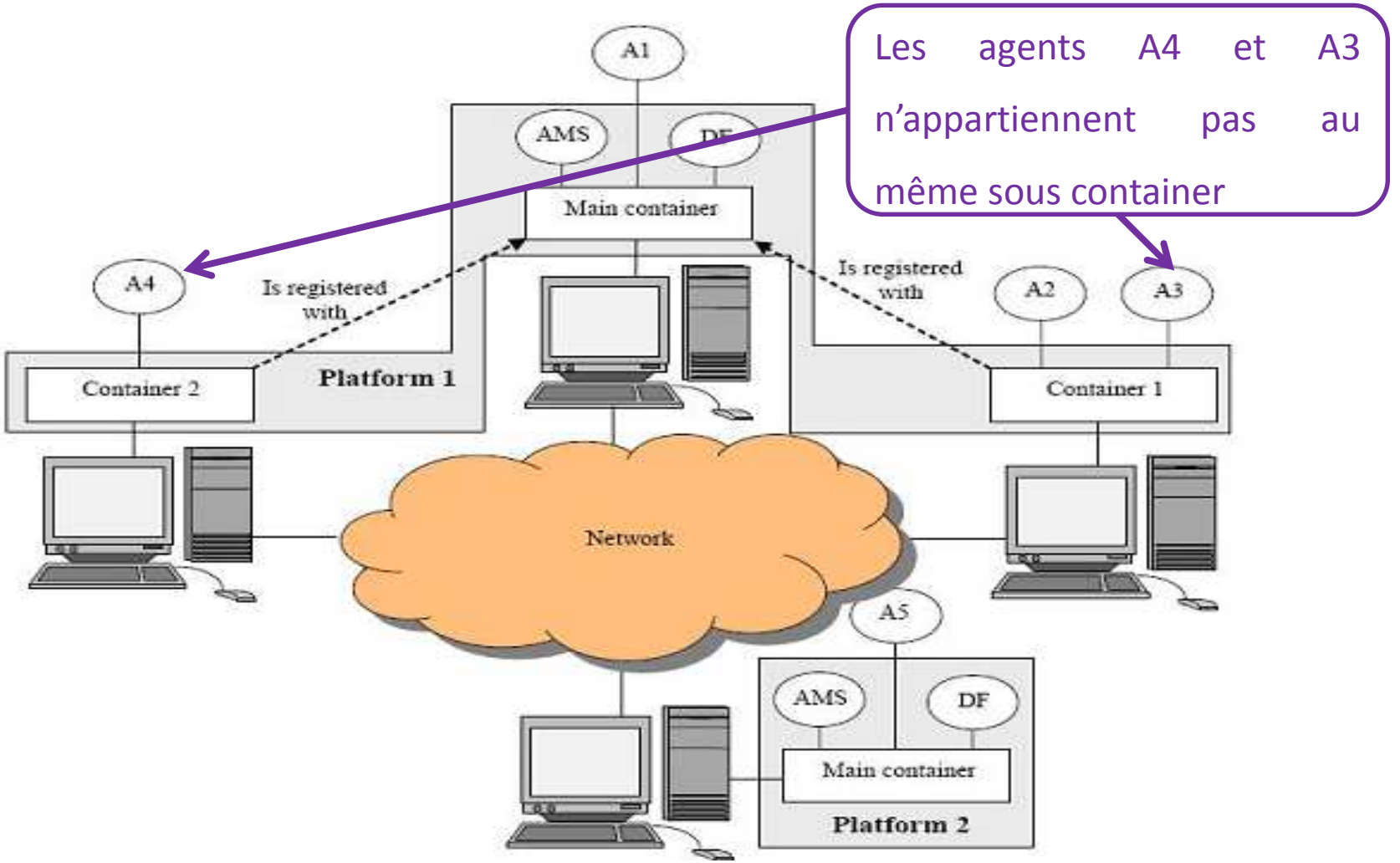
Introduction

Exemple de Containers (Environnement d'exécution pour les agents)



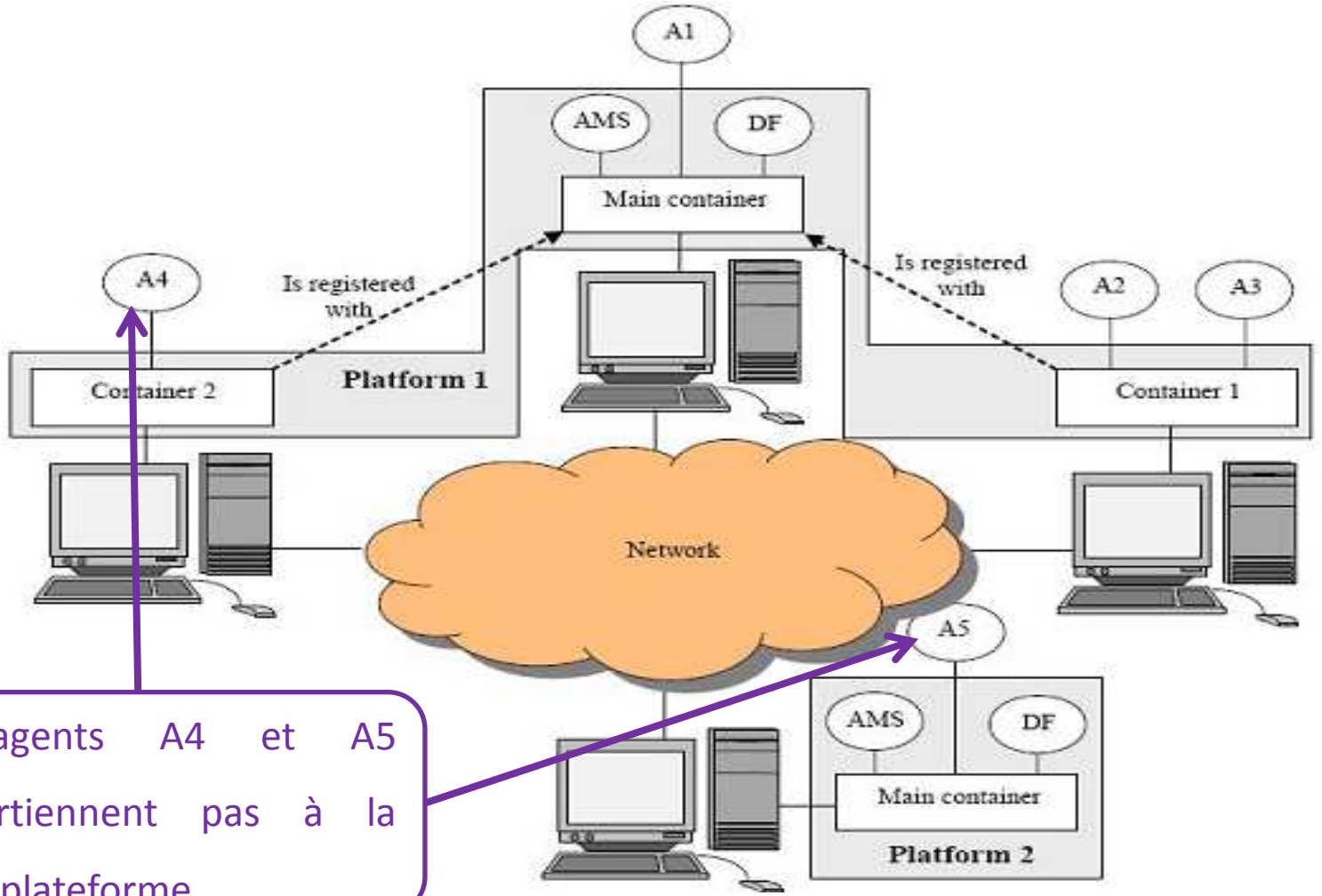
Introduction

Exemple de Containers (Environnement d'exécution pour les agents)



Introduction

Exemple de Containers (Environnement d'exécution pour les agents)



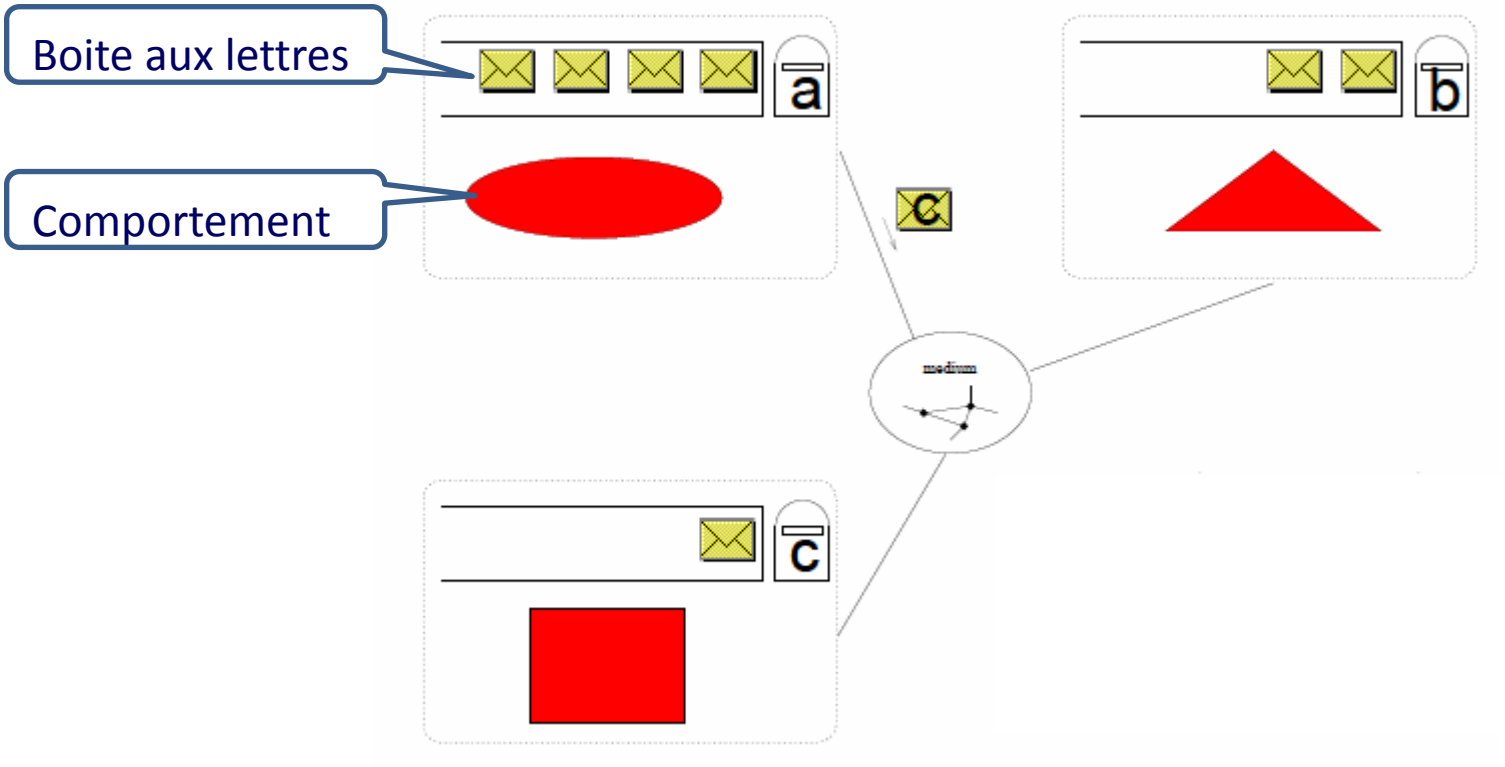
Les agents A4 et A5 n'appartiennent pas à la même plateforme

Système multi agents

- ✓ Un système multi agents est un système composé d'un ensemble d'agent.
- ✓ Un agent est une entité autonome réelle ou abstraite, possède un cycle de vie et peut **communiquer avec d'autres agents au moyen de messages,** les **messages** reçus sont stockés dans une **boîte aux lettres (file d'attente)**.
- ✓ Un agent possède des compétences et offres des services
- ✓ Un agent est caractérisé par un ou plusieurs **comportement** qui décrit la **réaction** de l'agent à un message reçu ou à une observation de ses connaissances. Ce **comportement** peut changer pendant son exécution.
- ✓ Un agent est identifié par un nom unique qui est l'**AgentIdentifier** (**AID**)
- ✓ Chaque agent peut joindre ou quitter librement la plateforme et rentrer en contact avec chacun des autres agents.

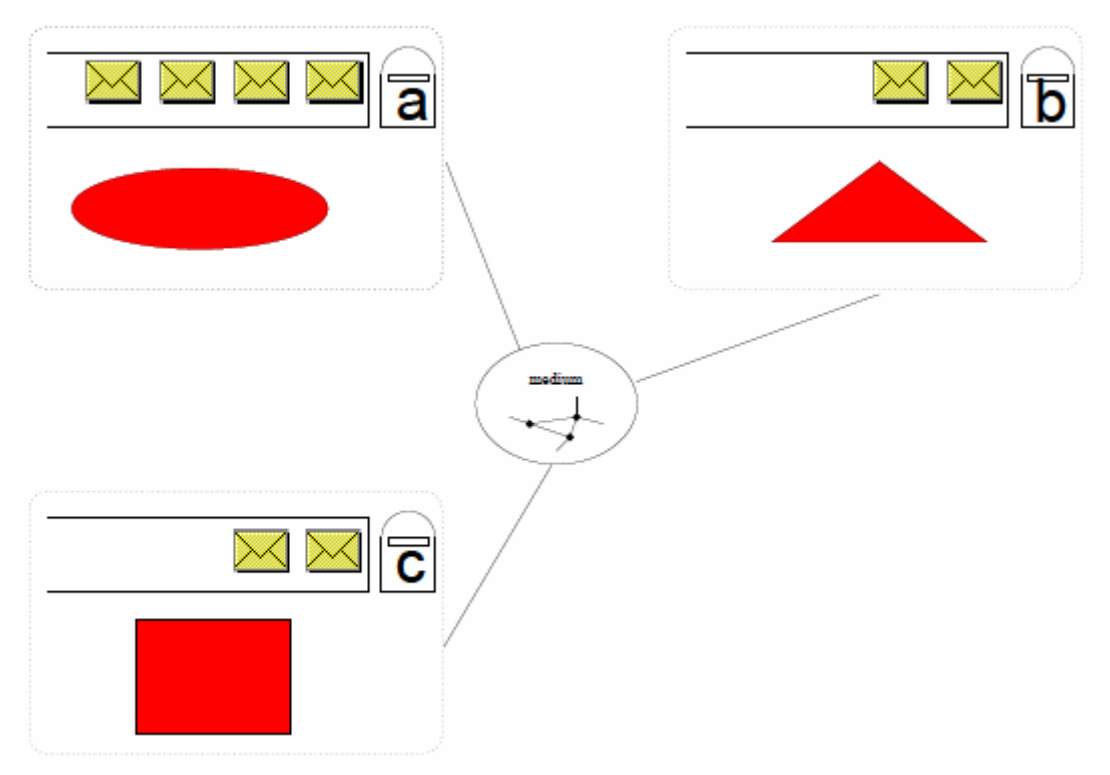
Système multi agents

- ✓ Dans cette exemple, nous avons 3 agents « a », « b » et « c »
- ✓ On suppose que l'agent « a » a envoyé un message à l'agent « c »



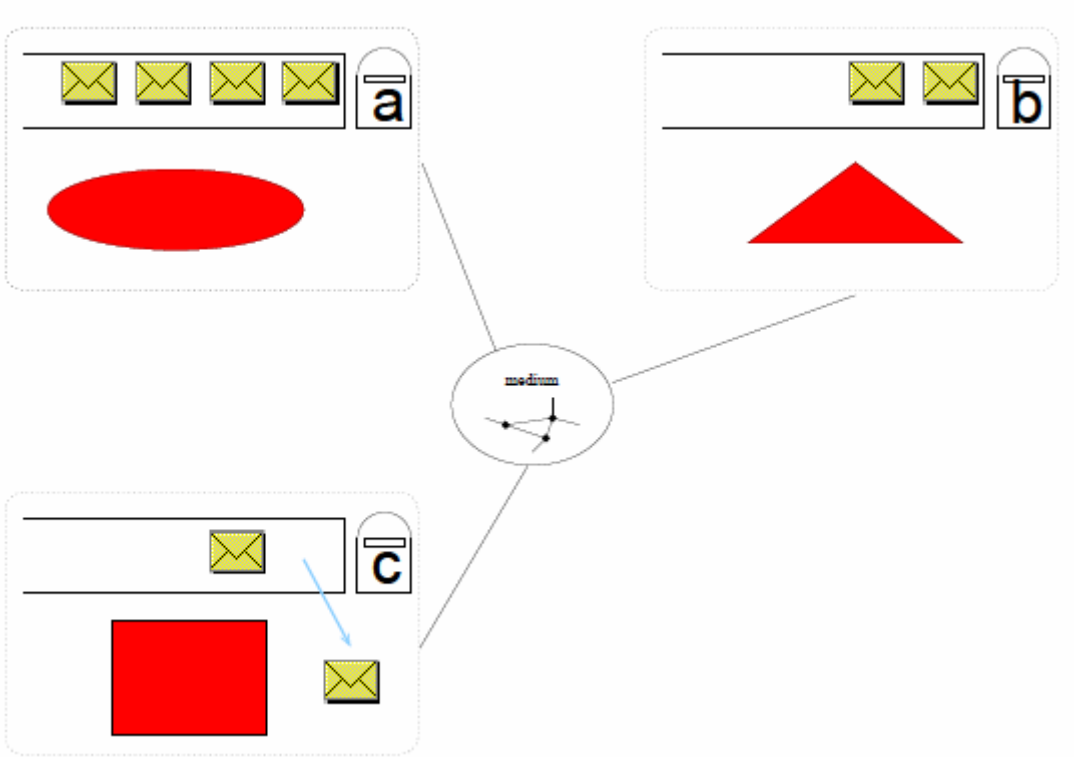
Système multi agents

- ✓ L'agent « c » reçoit le message et le stocke dans sa boite aux lettres



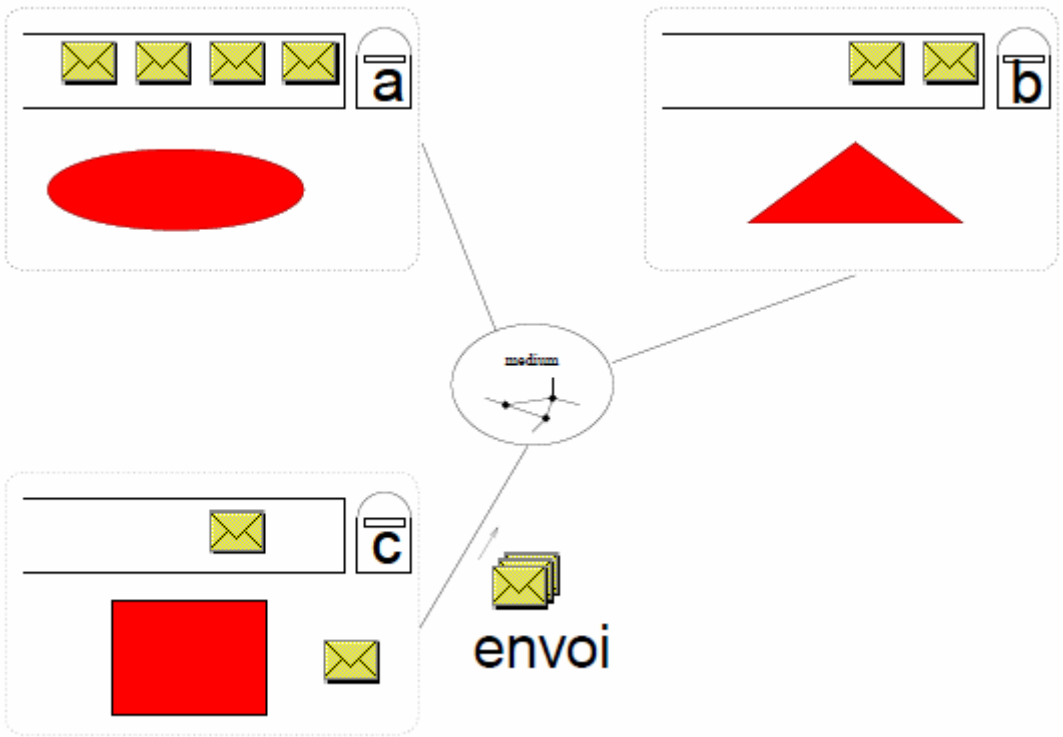
Système multi agents

- ✓ L'agent « c » consulte le 1er message de sa boite aux lettres



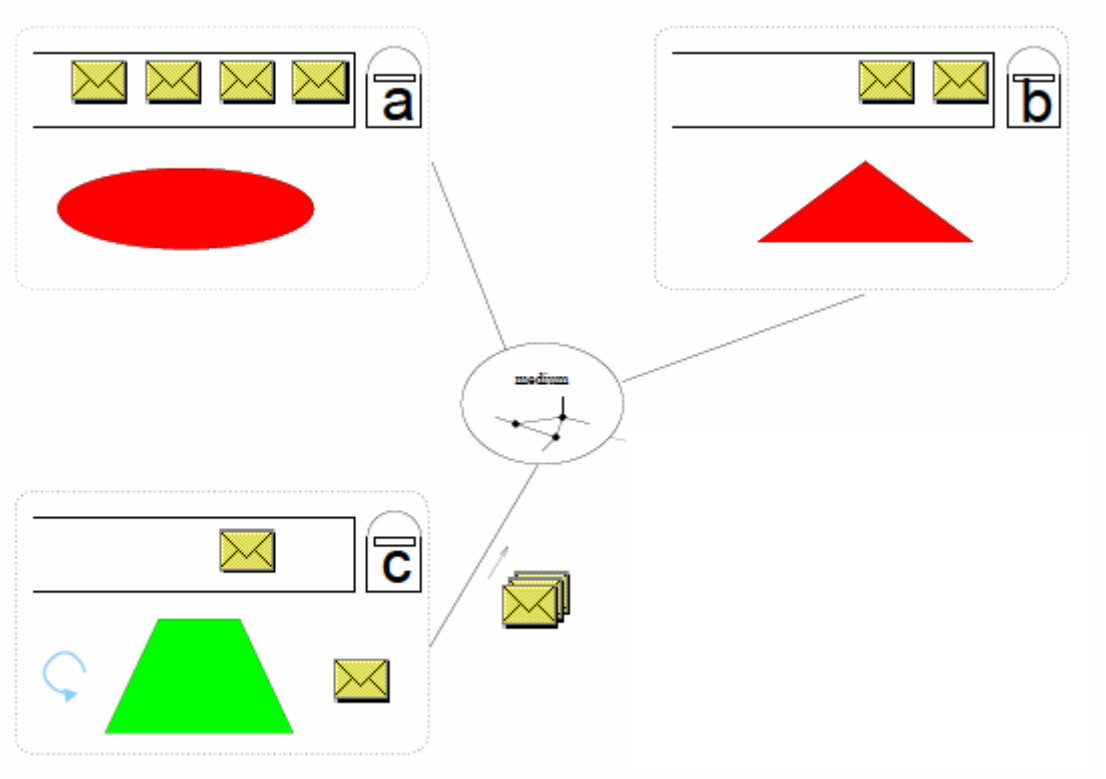
Système multi agents

- ✓ L'agent « c » après la consultation du 1er message, il peut
  - ➢ Envoyer des messages aux autres agents



Système multi agents

- ✓ L'agent « c » après la consultation du 1er message, il peut
  - ➢ Envoyer des messages aux autres agents
  - ➢ Changer son comportement



Installation de la plate forme JADE

- Télécharger le fichier JADE-all-4.5.0.zip à partir du site : **http ://www.jade.tilab.com**

- Décompresser le fichier JADE-all-4.5.0.zip.

  (Clique droit sur le nom du fichier puis cliquer sur **Extraire vers JADE-all-4.5.0**)

- Décompresser les fichiers (JADE-bin-4.5.0.zip, JADE-doc-4.5.0.zip, JADE-examples-4.5.0.zip, JADE-src-4.5.0.zip).

Création du premier agent avec JADE sous JAVA

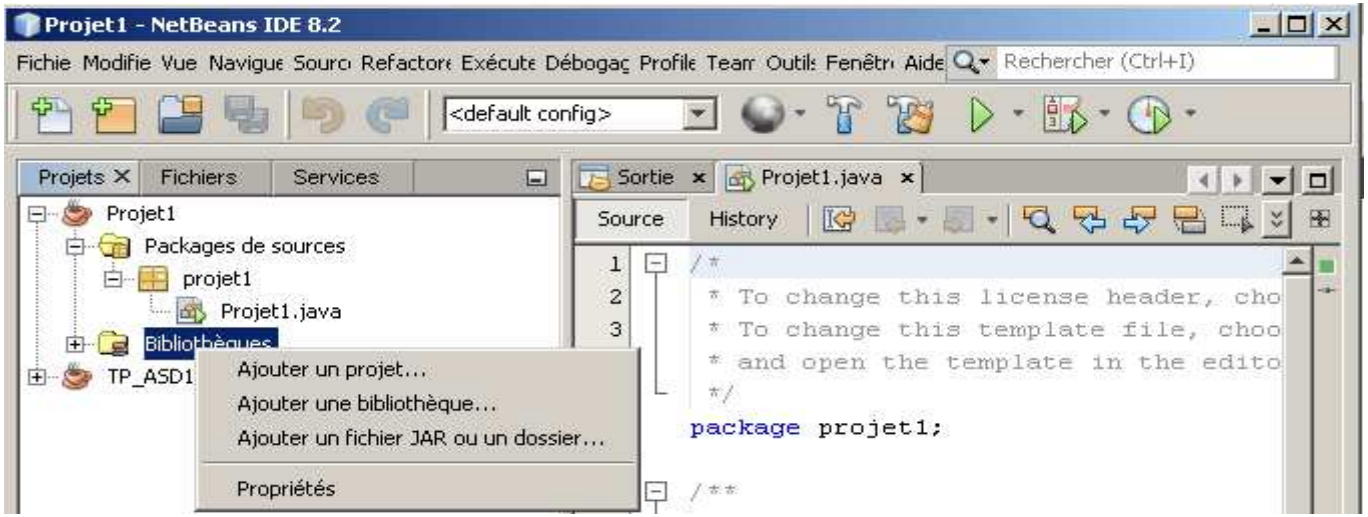
Sous Eclipse :

- Lancer Eclipse
- Créer un nouveau projet : **File/New/Java Project** (Projet1 par exemple)
- Cliquer sur **Next**, puis sur l'onglet **Libraries** , ensuite sur le bouton **Add External JARs**, puis rajouter le fichier jade.jar (**C:\JADE-all-4.5.0\JADE-bin-4.5.0\jade\lib**)
- Cliquer sur le bouton **Finish**
- Créer une nouvelle classe : **File/New/Class** (AgentTest par exemple)
- Cliquer sur le bouton **Finish**
- Taper le code suivant:

Création du premier agent avec JADE sous JAVA

Sous NetBeans :

- Lancer NetBeans
- Cliquer sur **Fichier/Nouveau projet…/Java**
- Cliquer sur **Suivant**
- Donner un nom à votre projet (**Projet1** par exemple)
- Dans l'arborescence du projet, faire une clique droit sur **Bibliothèques**



Création du premier agent avec JADE sous JAVA

Sous NetBeans :

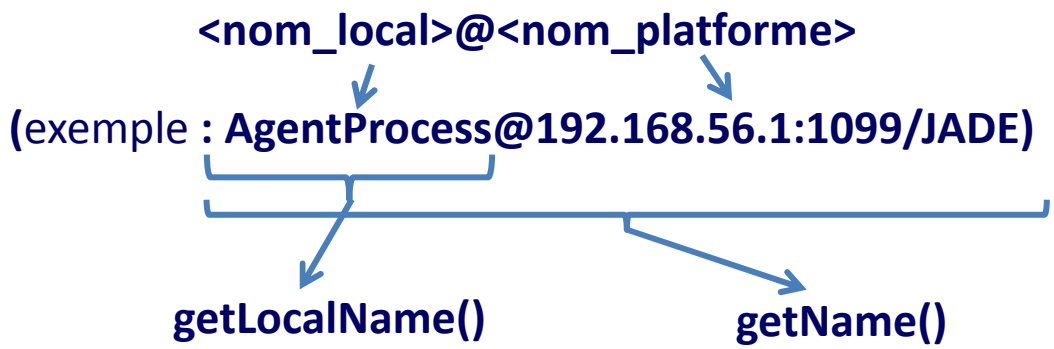
- Cliquer sur **Ajouter un fichier JAR ou un dossier**….
- Rajouter le fichier jade.jar (**C:\JADE-all-4[1].0.1\JADE-bin-4.5.0\jade\lib**)
- Cliquer sur le bouton **Ouvrir**
- Cliquer sur Créer une nouvelle classe : **Fichier/Nouveau fichier…./Java Class**
- Cliquer sur **Suivant**
- Donner un nom à votre classe (**AgentTest** par exemple)
- Cliquer sur le bouton **Terminer**
- Taper le code suivant:

Création du premier agent avec JADE et ECLIPSE

```
import jade.core.Agent;

public class AgentTest extends Agent{ //l'agent hérite de la classe jade.core.Agent

        protected void setup() {

                System.out.println("Je suis l'agent : "+this.getLocalName());
                System.out.println("Je suis l'agent : "+this.getName());

        }

}
```

Création du premier agent avec JADE sous JAVA

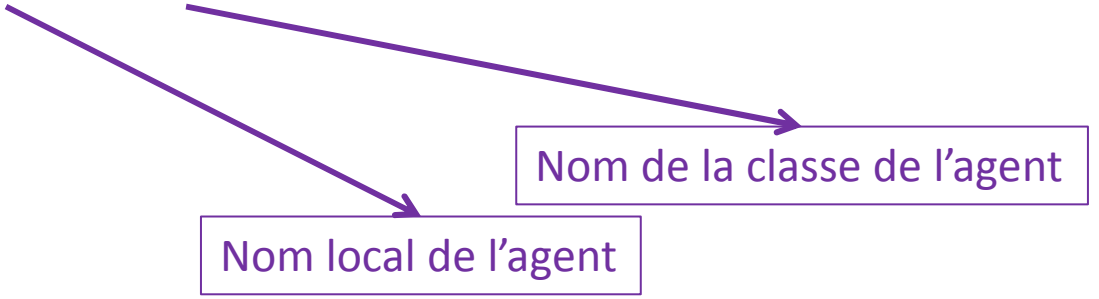
- ✓Chaque agent hérite de la classe **jade.core.Agent**
- ✓Chaque agent doit avoir une méthode obligatoire **void setup**() dans laquelle, il :
  - •Récupère les paramètres passés en arguments
  - •Enregistre les Services offerts par l'agent auprès du DF
  - •Lance ses comportements (behaviors)
- ✓Un agent est identifié par un nom unique l'**AgentIdentifier** (**AID**)
- ✓Un **AID** a la forme suivante :



Création du premier agent avec JADE sous JAVA

- Pour tester votre premier agent, taper le code suivant :

```
public class Test {
   public static void main(String[] args) {
            String [] commande = new String[3];
            String argument ="";
            argument = argument+"agent1:AgentTest";
            commande [0]="-cp";
            commande [1]="jade.boot";
            commande [2]= argument;
            jade.Boot.main(commande);
   }
}
```



- Cliquer sur **Run** pour exécuter la méthode main de la classe Test
- **Remarque 1** : Si vous utilisez un package alors utilisez l'instruction suivante :

argument = argument+"agent1:**NomPackage.**AgentTest";

Création du premier agent avec JADE sous JAVA

- **Remarque 2** : Si vous voulez lancer plusieurs agent avec la même classe alors utilisez le code suivant :

```
public class Test {
   public static void main(String[] args) {
            String [] commande = new String[3];
            String argument ="";
            argument = argument+"agent1:AgentTest";
            argument = argument+";";
            argument = argument+"agent2:AgentTest";
            commande [0]="-cp";
            commande [1]="jade.boot";
            commande [2]= argument;
            jade.Boot.main(commande);
   }
}
```

- Cliquer sur **Run** pour exécuter la méthode main de la classe Test

Création du premier agent avec JADE sous JAVA

✓ **Récupérer les paramètres passés en arguments :**

•Dans la plateforme JADE, nous pouvons faire passer des paramètres aux agents.

• Chaque agent récupère ses paramètre en utilisant la méthode **getArguments()**.

• **Exemple :**

```
import jade.core.Agent;
public class AgentTest  extends Agent{
        protected void setup() {
                System.out.println("Je suis l'agent:"+this.getLocalName());
                System.out.println(" Mes arguments sont :");
                Object[] args = getArguments();
                if (args != null) {
                        for (int i = 0; i < args.length; ++i) {
                                System.out.println(args[i]);
                        }
                }
        }
}
```

Création du premier agent avec JADE sous JAVA

✓ **Récupérer les paramètres passés en arguments :**

1. Tester le programme par la classe Test suivante :

```
public class Test {
   public static void main(String[] args) {
            String [] commande = new String[3];
            String argument ="";
            argument = argument+"agent1:AgentTest(TP,-45,30) ";
            commande [0]="-cp";
            commande [1]="jade.boot";
            commande [2]= argument;
            jade.Boot.main(commande);
   }
}
```

Création du premier agent avec JADE sous JAVA

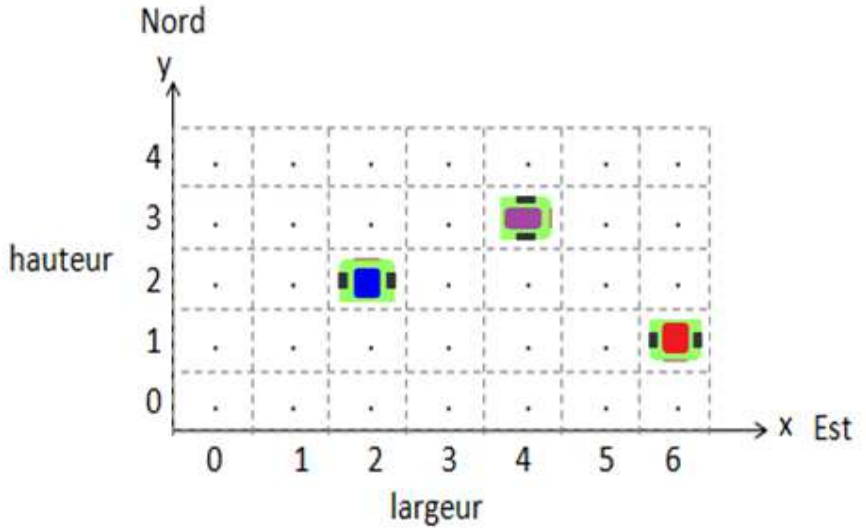
✓ **Récupérer les paramètres passés en arguments :**

2. Tester le programme par la classe Test suivante :

```
public class Test {
   public static void main(String[] args) {
            String [] commande = new String[3];
            String argument ="";
            argument = argument+"agent1:AgentTest(TP,-45,30) ";
            argument = argument+";";
            argument = argument+"agent2:AgentTest(ALDI,5,13)";
            commande [0]="-cp";
            commande [1]="jade.boot";
            commande [2]= argument;
            jade.Boot.main(commande);
   }
}
```

Test de connaissances

Nous désirons développer une application pour simuler le comportement des robots mobiles sur un plan orthonormé à 4 directions. Un robot sera considéré comme un agent qui reçoit dans sa liste de paramètres les données suivantes : les dimensions (hauteur, largeur) du plan de déplacement, la position initiale (donnée par les coordonnées x et y) et une direction (indiquée par l'une des valeurs : « Nord », « Sud », « Est » et « Ouest »). Chaque robot doit récupérer ses paramètres et les convertir selon un format adéquat pour les stocker dans des variables puis les afficher.



Test de connaissances

Travail demandé :

1.Proposer une implémentation de la classe Robot

2.Tester la classe Robot en utilisant 1 Robot puis 2 Robots

Exemple de Robot avec paramètres :

Robucar:Robot(15,20,7,10,Est)

Atlas:Robot (15,20,4,3,Nord);NAO:AgentTest(15,20,12,5,Sud)

Remarque:

Pour transformer une chaine de caractères en un nombre entier, utiliser la méthode Integer.parseInt(…).

Exemple

```
String st = "123";
int val = Integer.parseInt(st);
```

Test de connaissances

Test avec un seul Robot :

```
public class Test {
   public static void main(String[] args) {
            String [] commande = new String[3];
            String argument ="";
            argument = argument+"Robucar:Robot(15,20,7,10,Est)";
            commande [0]="-cp";
            commande [1]="jade.boot";
            commande [2]= argument;
            jade.Boot.main(commande);
   }
}
```

Test de connaissances

Test avec 2 Robots :

```
public class Test {
   public static void main(String[] args) {
            String [] commande = new String[3];
            String argument ="";
            argument = argument+"Atlas:Robot(15,20,4,3,Nord)";
            argument = argument+";";
            argument = argument+"NAO: Robot(15,20,12,5,Sud)";
            commande [0]="-cp";
            commande [1]="jade.boot";
            commande [2]= argument;
            jade.Boot.main(commande);
   }
}
```

Test de connaissances

Date de la remise : 26/02/2022

Le travail à remettre doit contenir :

- **La classe Robot (fichier .java) et**
- **Les captures d'écran des 2 exécutions.**

Il faut les compresser dans un seul fichier qui porte un nom de la forme : TPALDI01_NomEtudiant1_NomEtudiant2

Le travail doit être envoyé l'adresse émail : lamia.mezai@univ-constantine2.dz