

Applications Réparties

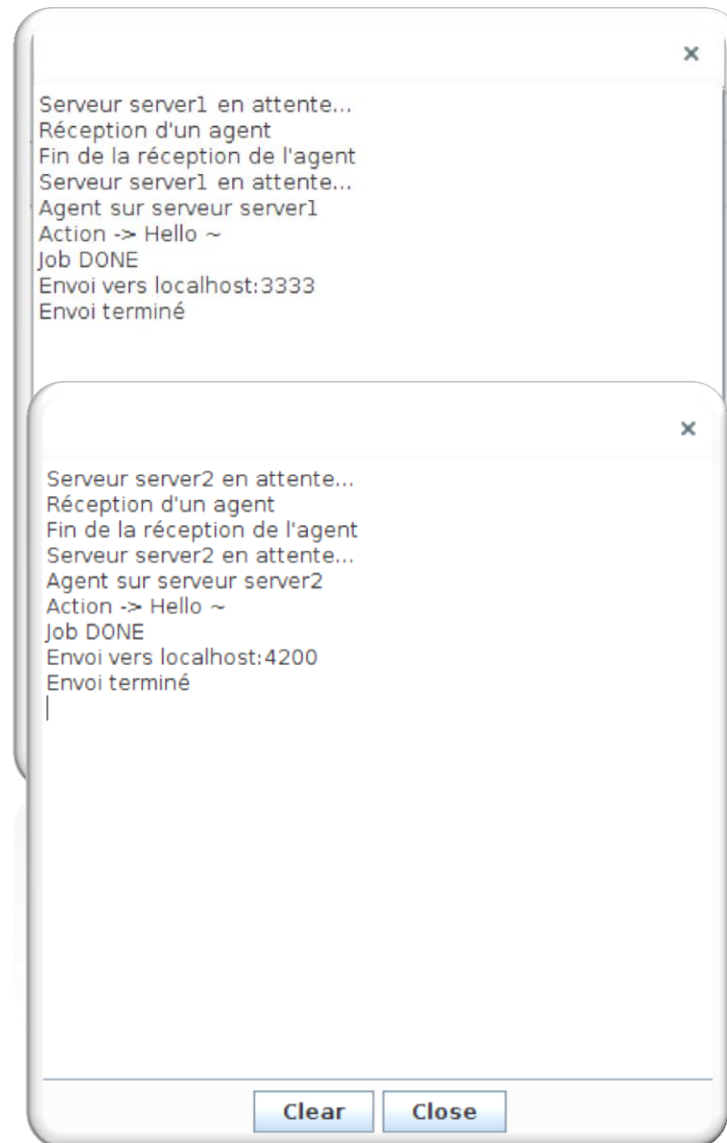
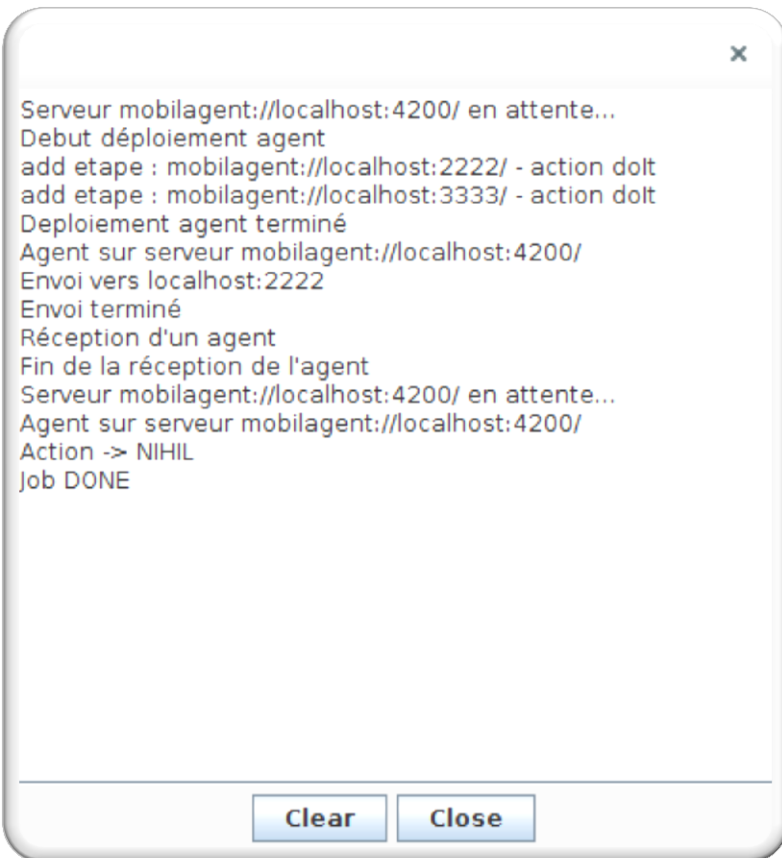
Agents Mobiles - RMI

EUDES Robin – ROSSI Ombeline

SOMMAIRE

- 1. Agent Hello**
- 2. Agent LookForHotel**
- 3. LookForHotel – version RMI**
- 4. Comparaison des solutions**
- 5. Ajout d'un annuaire de courtage**
- 6. Bilan**

Agent Hello



Boucle d'exécution

```
if(todo){
    route.next().get_action().execute();
    Starter.get_logger().log(Level.FINE,"Job DONE");
}

if(route.hasNext()){
    try {
        Starter.get_logger().log(Level.FINE,"Envoi vers "+ route.get().server.getHost()+":"+route.get().

        todo=true;

        // construction du socket
        Socket socket_agent = new Socket(route.get().server.getHost(),route.get().server.getPort());

        // construction de l'output stream
        ObjectOutputStream output = new ObjectOutputStream(socket_agent.getOutputStream());

        // envoi du repository de l'agent
        output.writeObject(bma.getjar());
        // envoi de l'agent
        output.writeObject(this);

        output.close();
        socket_agent.close();

        Starter.get_logger().log(Level.FINE,"Envoi terminé");

    } catch (Exception e) {
```

Boucle de réception

```
try {  
    ServerSocket socket = new ServerSocket(port);  
  
    while(alive){  
        Starter.get_logger().log(Level.FINE,"Serveur "+this.get_srv_name()+ " en attente...");  
        // On accepte la connexion  
        Socket agent = socket.accept();  
        Starter.get_logger().log(Level.FINE,"Réception d'un agent");  
  
        // prep Agent Loader  
        BAMAgentClassLoader bma = new BAMAgentClassLoader(new URL[] {}, bms);  
  
        // On prépare la réception de l'agent  
        InputStream input = agent.getInputStream();  
        AgentInputStream agent_in = new AgentInputStream(input, bma);  
  
        // On reçoit le jar des méthodes de l'agent  
        Jar jarfile = (Jar) agent_in.readObject();  
        agent_in.loader.extractCode(jarfile);  
  
        // On reçoit l'agent & on l'init.  
        Agent agentob = (Agent) agent_in.readObject();  
        agentob.init(agent_in.loader, this, this.get_srv_name());  
  
        // Fin de la conec  
        agent_in.close();  
  
        Starter.get_logger().log(Level.FINE,"Fin de la réception de l'agent");  
  
        // On lance l'agent !  
        new Thread(agentob).start();  
    }  
    socket.close();  
}
```

LookForHotel – get_hotels

```
/**
 * On peut interroger plusieurs serveurs pour avoir une liste d'hotels
 */
protected _Action get_hotels = new _Action(){

    private static final long serialVersionUID = 8258631604519690491L;

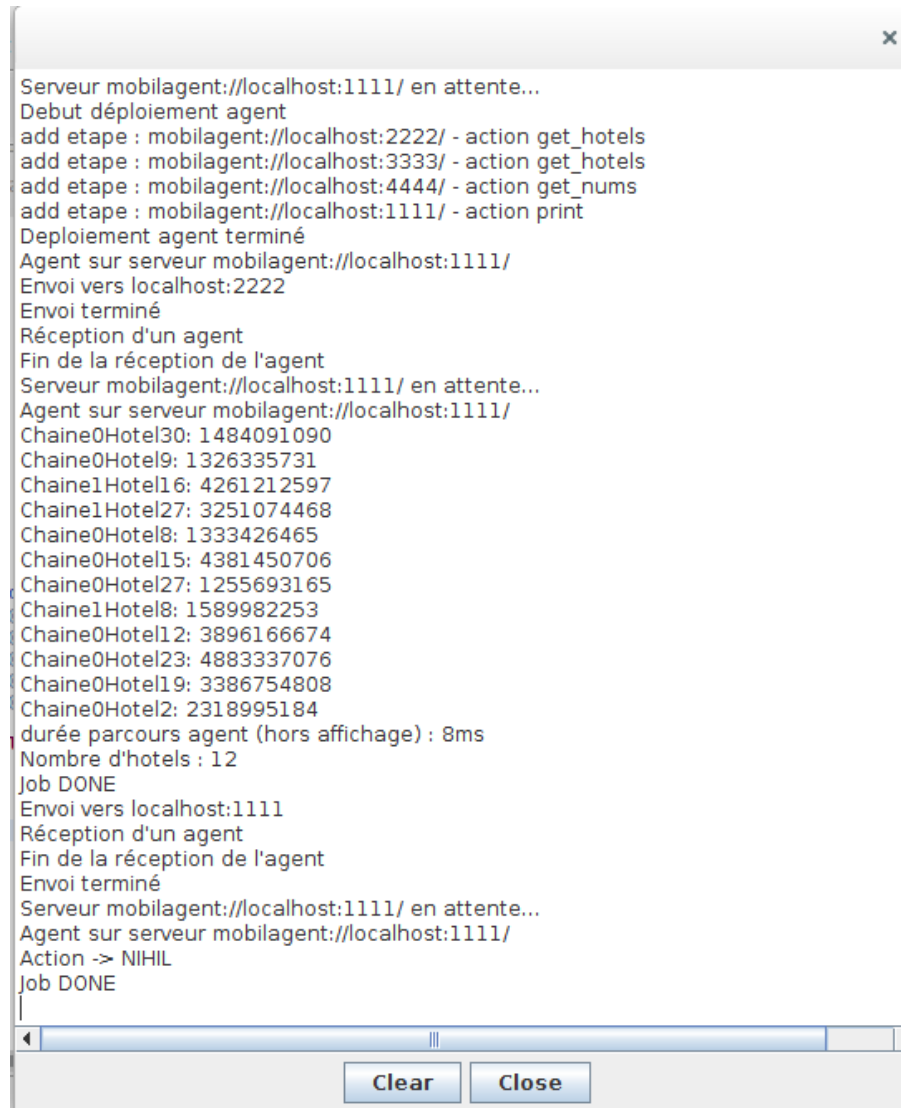
    @SuppressWarnings("unchecked")
    @Override
    public void execute() {
        _Service<List<Hotel>> service = (_Service<List<Hotel>>) srv.getService("Hotels");
        hotels.addAll(service.call(localisation));
    }
};
```

Service - Hotels

```
/**
 * Retourne la liste d'hotels matchant la loc
 */
@Override
public List<Hotel> get(String localisation) {
    List<Hotel> l = new LinkedList<Hotel>();
    for(Hotel h:hotels){
        if(h.localisation.equalsIgnoreCase(localisation)){
            l.add(h);
        }
    }
    return l;
}

/**
 * Action effectuée à l'appel du service
 * Retourne le resultat de get
 * un seul param, la loc
 */
@Override
public List<Hotel> call(Object... params) throws IllegalArgumentException {
    return get(params[0].toString());
}
```

Agent LookForHotel

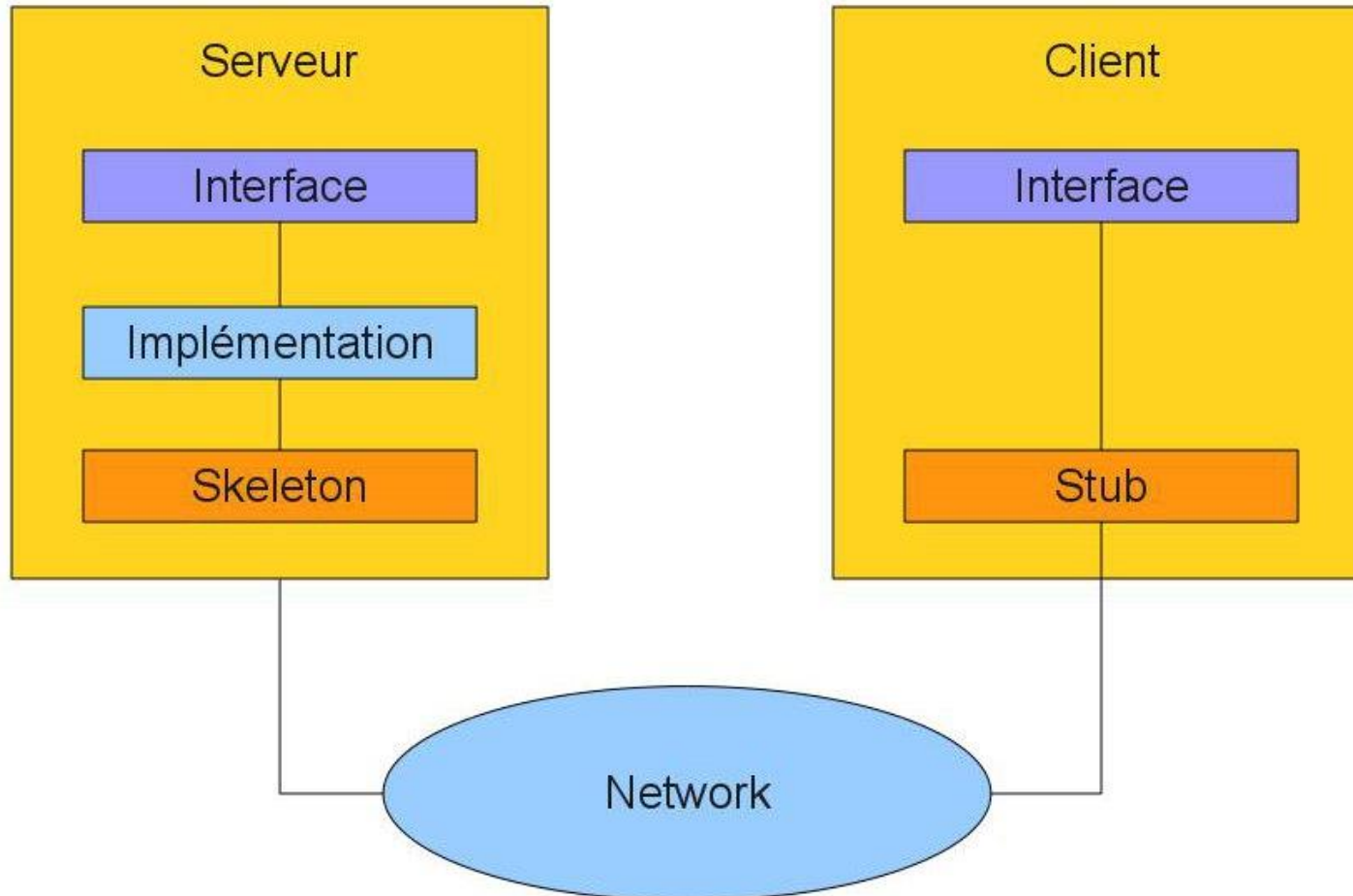


```

Serveur mobilagent://localhost:1111/ en attente...
Debut déploiement agent
add etape : mobilagent://localhost:2222/ - action get_hotels
add etape : mobilagent://localhost:3333/ - action get_hotels
add etape : mobilagent://localhost:4444/ - action get_nums
add etape : mobilagent://localhost:1111/ - action print
Déploiement agent terminé
Agent sur serveur mobilagent://localhost:1111/
Envoi vers localhost:2222
Envoi terminé
Réception d'un agent
Fin de la réception de l'agent
Serveur mobilagent://localhost:1111/ en attente...
Agent sur serveur mobilagent://localhost:1111/
Chaine0Hotel30: 1484091090
Chaine0Hotel9: 1326335731
Chaine1Hotel16: 4261212597
Chaine1Hotel27: 3251074468
Chaine0Hotel8: 1333426465
Chaine0Hotel15: 4381450706
Chaine0Hotel27: 1255693165
Chaine1Hotel8: 1589982253
Chaine0Hotel12: 3896166674
Chaine0Hotel23: 4883337076
Chaine0Hotel19: 3386754808
Chaine0Hotel2: 2318995184
durée parcours agent (hors affichage) : 8ms
Nombre d'hotels : 12
Job DONE
Envoi vers localhost:1111
Réception d'un agent
Fin de la réception de l'agent
Envoi terminé
Serveur mobilagent://localhost:1111/ en attente...
Agent sur serveur mobilagent://localhost:1111/
Action -> NIHIL
Job DONE

```


LookForHotel – version RMI



RMI – Server Side

2 interfaces `_Annuaire` et `_Chaine`.

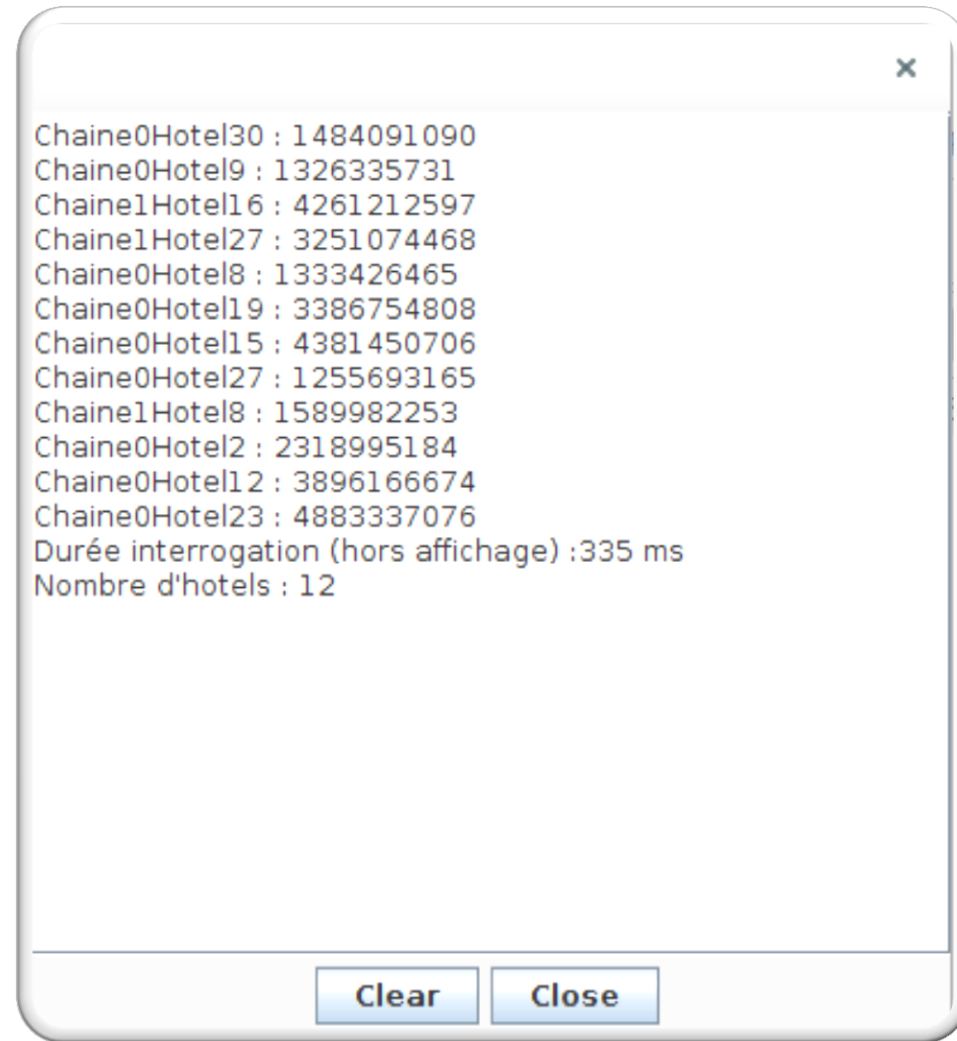
Implémentation des interfaces côté serveur

```
try {
    if(args[0].equalsIgnoreCase("Chaine")){
        _Chaine skeleton;
        // exportobject déjà réalisé car la classe extends UnicastRemoteObject
        skeleton = (_Chaine)new Chaine(args[1]);
        Registry registry = LocateRegistry.createRegistry(Integer.parseInt(args[2]));
        registry.rebind("Chaine",skeleton);
        logger.log(Level.FINE,"Service Chaine disponible");
    }else if(args[0].equalsIgnoreCase("Annuaire")){
        _Annuaire skeleton = (_Annuaire)new Annuaire(args[1]); // idem
        Registry registry = LocateRegistry.createRegistry(Integer.parseInt(args[2]));
        registry.rebind("Annuaire",skeleton);
        logger.log(Level.FINE,"Service Annuaire disponible");
    }else{
        logger.log(Level.WARNING,"Service demandé inconnu");
    }
} catch (ParserConfigurationException | SAXException | IOException e) {
    logger.log(Level.WARNING,e.toString());
}
```

RMI – Client Side

```
/**
 * On peut contacter n chaines d'hotels, décrite par 4 params : <serveur name> <serveur port> "Chaine" <localisation>
 * Et on termine par 1 unique annuaire décrit par 3 params : <serveur name> <serveur port> "Annuaire"
 */
while(count<args.length){
    try {
        if(args[count+2].equalsIgnoreCase("Chaine")){
            Registry registry;
            registry = LocateRegistry.getRegistry(InetAddress.getByName(args[count]).getHostAddress());
            _Chaine stub = (_Chaine) registry.lookup(args[count+2]);
            hotels.addAll(stub.get(args[count+3]));
            count+=4;
        }else if(args[count+2].equalsIgnoreCase("Annuaire")){
            Registry registry = LocateRegistry.getRegistry(InetAddress.getByName(args[count]).getHostAddress());
            _Annuaire stub = (_Annuaire) registry.lookup(args[count+2]);
            for(Hotel h:hotels){
                annuaire.put(h.name,stub.get(h.name));
            }
            count+=3;
        }
    } catch (NumberFormatException | RemoteException
            | UnknownHostException | NotBoundException e ) {
        logger.log(Level.WARNING,e.toString());
    }
}
```

RMI – Client Side



Comparaison des solutions

BAM

- Recherche des Hotels de Paris - 3 serveurs (chaine 1 , 2 + annuaire)
 - durée parcours agent (hors affichage) : 5ms
 - Nombre d'hotels : 19991

- Recherche des Hotels de Grenoble - 3 serveurs (chaine 1 , 2 + annuaire)
 - durée parcours agent (hors affichage) : 2ms
 - Nombre d'hotels : 10

- **La durée du service ne semble pas être impactée par le critère de recherche, les variations autour de la durée dépendent plus de l'utilisation actuelle de la carte réseau par l'OS qu'autre chose.**

Comparaison des solutions

RMI

- Recherche des Hôtels de Paris - 3 serveurs (chaine 1 , 2 + annuaire)
 - Durée interrogation (hors affichage) :3656 ms
 - Nombre d'hôtels : 19991
- Recherche des Hôtels de Grenoble - 3 serveurs (chaine 1 , 2 + annuaire)
 - Durée interrogation (hors affichage) :319 ms
 - Nombre d'hôtels : 10
- **La durée du service est directement impactée par la taille de la liste d'hôtels à rechercher dans l'annuaire. En effet, pour chaque hôtel -> un appel RMI au service Annuaire...**

Ajout d'un annuaire de courtage

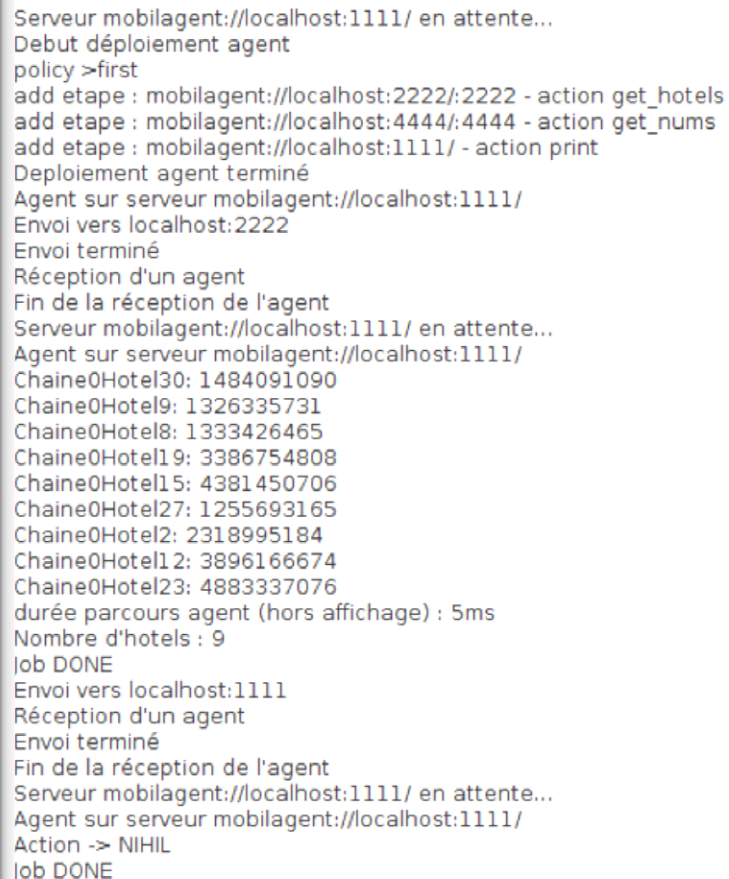
- Serveur RMI d'annuaire
- Ajout pour le service d'une entrée dans la Map
- Un agent choisit juste une politique, pas les serveurs qu'il interrogera.

Interrogation du serveur RMI

```
@Override
public LinkedList<URI> getservice(String name) throws RemoteException {
    LinkedList<URI> tmp = new LinkedList<URI>();
    for(Regfield r:registre.get(name)){
        if(r.av){
            tmp.add(r.serveur);
        }
    }
    return tmp;
}

@Override
public void registerservice(String name, URI src, boolean availability)
    throws RemoteException {
    /**
     * Si le service n'a jamais été add, on crée l'entrée
     */
    if(!registre.containsKey(name)){
        registre.put(name,new LinkedList<Regfield>());
    }
    // On ajoute le serveur proposant ce service à la liste
    registre.get(name).add(new Regfield(src,availability));
}
```


Policy -> First

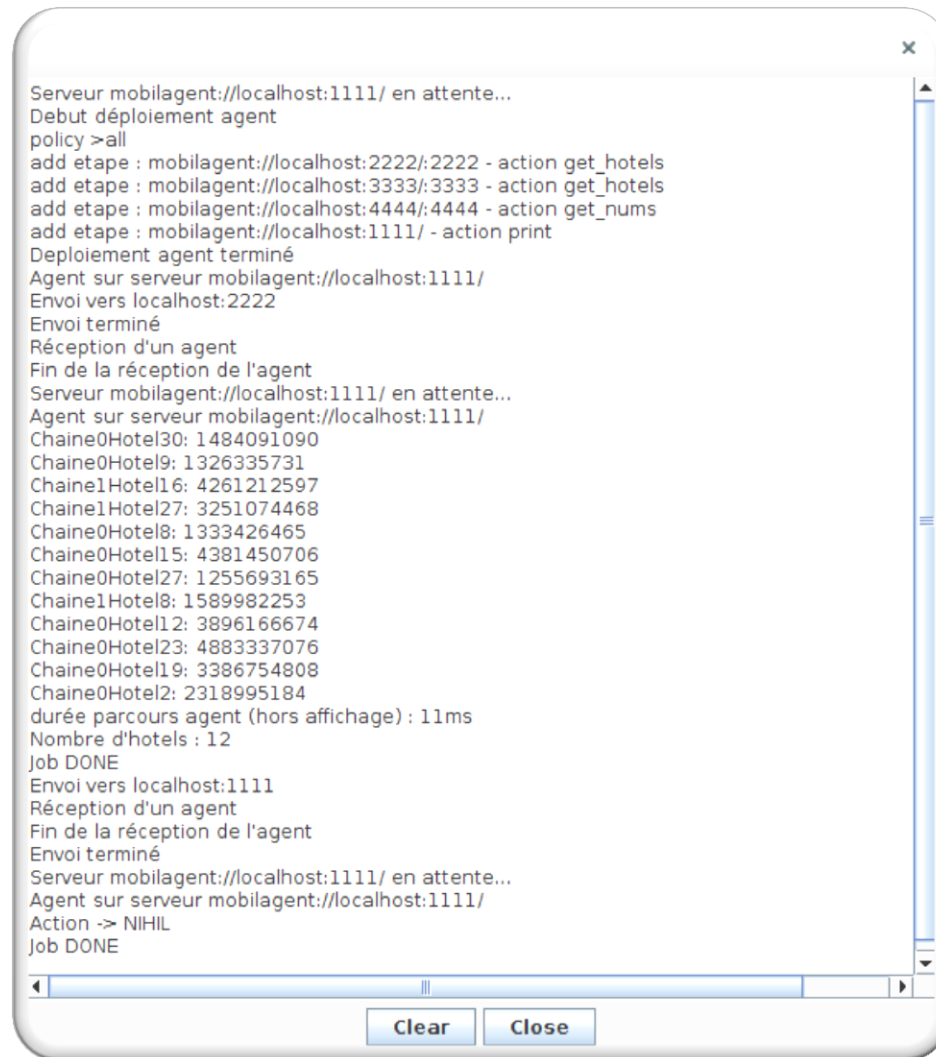


Server mobilagent://localhost:1111/ en attente...
Debut déploiement agent
policy >first
add etape : mobilagent://localhost:2222/:2222 - action get_hotels
add etape : mobilagent://localhost:4444/:4444 - action get_nums
add etape : mobilagent://localhost:1111/ - action print
Déploiement agent terminé
Agent sur serveur mobilagent://localhost:1111/
Envoi vers localhost:2222
Envoi terminé
Réception d'un agent
Fin de la réception de l'agent
Server mobilagent://localhost:1111/ en attente...
Agent sur serveur mobilagent://localhost:1111/
Chaine0Hotel30: 1484091090
Chaine0Hotel9: 1326335731
Chaine0Hotel8: 1333426465
Chaine0Hotel19: 3386754808
Chaine0Hotel15: 4381450706
Chaine0Hotel27: 1255693165
Chaine0Hotel2: 2318995184
Chaine0Hotel12: 3896166674
Chaine0Hotel23: 4883337076
durée parcours agent (hors affichage) : 5ms
Nombre d'hotels : 9
Job DONE
Envoi vers localhost:1111
Réception d'un agent
Envoi terminé
Fin de la réception de l'agent
Server mobilagent://localhost:1111/ en attente...
Agent sur serveur mobilagent://localhost:1111/
Action -> NIHIL
Job DONE

Clear

Close

Policy -> All



```

Serveur mobilagent://localhost:1111/ en attente...
Debut déploiement agent
policy >all
add etape : mobilagent://localhost:2222/2222 - action get_hotels
add etape : mobilagent://localhost:3333/3333 - action get_hotels
add etape : mobilagent://localhost:4444/4444 - action get_nums
add etape : mobilagent://localhost:1111/ - action print
Déploiement agent terminé
Agent sur serveur mobilagent://localhost:1111/
Envoi vers localhost:2222
Envoi terminé
Réception d'un agent
Fin de la réception de l'agent
Serveur mobilagent://localhost:1111/ en attente...
Agent sur serveur mobilagent://localhost:1111/
Chaine0Hotel30: 1484091090
Chaine0Hotel9: 1326335731
Chaine1Hotel16: 4261212597
Chaine1Hotel27: 3251074468
Chaine0Hotel8: 1333426465
Chaine0Hotel15: 4381450706
Chaine0Hotel27: 1255693165
Chaine1Hotel8: 1589982253
Chaine0Hotel12: 3896166674
Chaine0Hotel23: 4883337076
Chaine0Hotel19: 3386754808
Chaine0Hotel2: 2318995184
durée parcours agent (hors affichage) : 11ms
Nombre d'hotels : 12
Job DONE
Envoi vers localhost:1111
Réception d'un agent
Fin de la réception de l'agent
Envoi terminé
Serveur mobilagent://localhost:1111/ en attente...
Agent sur serveur mobilagent://localhost:1111/
Action -> NIHIL
Job DONE

```

Clear Close

Bilan

- RMI « bien plus rapide » à mettre en œuvre, plus souple. Mais utilisation du réseau plus importante.
- BAM plus complexe, moins souple, mais préférable pour des calculs lourds

Bilan

- Un projet qui nous aura permis de comprendre en détail les notions abordées ce semestre