TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ASE-9426 Distributed Automation Systems Design

Assignment 2: Design and implementation of a Multi-Agent System in JADE

Submitted by

Md Aman Khan, ID:272541

Junaid Iqbal, ID:272538

Introduction

A multi-agent system is a loosely coupled network of problem-solver entities that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity. This assignment related to multiple problem-solving entities ideally suited for Multi-agent systems.

Objectives

- Understanding MAS principles
- Ability to model and solve problems using a MAS methodology
- Learn the usage of JADE
- Learn agents behaviors
- Learn basic FIPA concepts
- Understand the importance of system reconfigurability
- Get familiar with intelliJ and JAVA programming

Requirements

- Java Agent Development Framework (JADE)
- intelliJ
- MAS methodology
- UML Modelling tools

Task1

Route finder

Task1 is to create a MAS using the methodology given in the lecture which is capable to find the shortest route in order to move a pallet between end points in a conveyor system. The overview of the methodology is shown below:

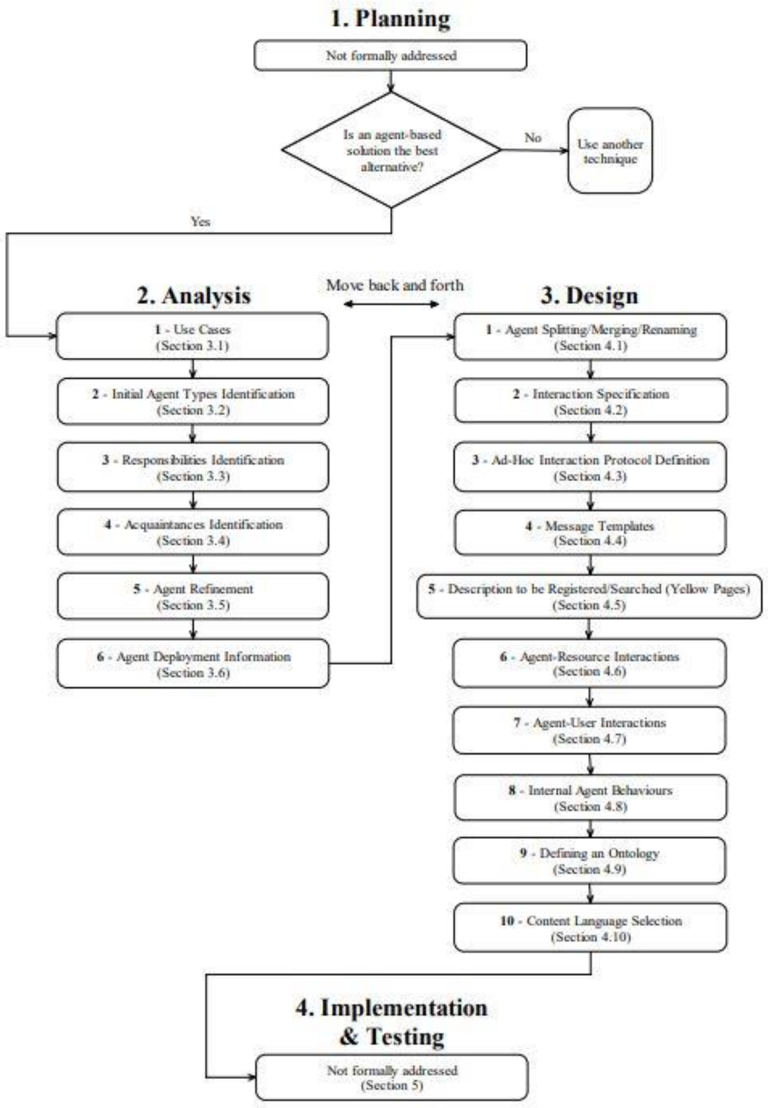

1. Planning
Not formally addressed
Is an agent-based solution the best alternative?
No
Use another technique
Yes
2. Analysis
Move back and forth
3. Design
1 - Use Cases (Section 3.1)
2 - Initial Agent Types Identification (Section 3.2)
3 - Responsibilities Identification (Section 3.3)
4 - Acquaintances Identification (Section 3.4)
5 - Agent Refinement (Section 3.5)
6 - Agent Deployment Information (Section 3.6)
1 - Agent Splitting/Merging/Renaming (Section 4.1)
2 - Interaction Specification (Section 4.2)
3 - Ad-Hoc Interaction Protocol Definition (Section 4.3)
4 - Message Templates (Section 4.4)
5 - Description to be Registered/Searched (Yellow Pages) (Section 4.5)
6 - Agent-Resource Interactions (Section 4.6)
7 - Agent-User Interactions (Section 4.7)
8 - Internal Agent Behaviours (Section 4.8)
9 - Defining an Ontology (Section 4.9)
10 - Content Language Selection (Section 4.10)
4. Implementation & Testing
Not formally addressed (Section 5)

Figure 1: Overview of the methodology

1. Planning

Task 1 is related to the patterns reviewing. Additionally, MAS is suited for distributed components, System complex behavior by targeting/thinking from the component behavior. So, Agent base solution is the best.

2.Analysis

Step 1: Use Cases

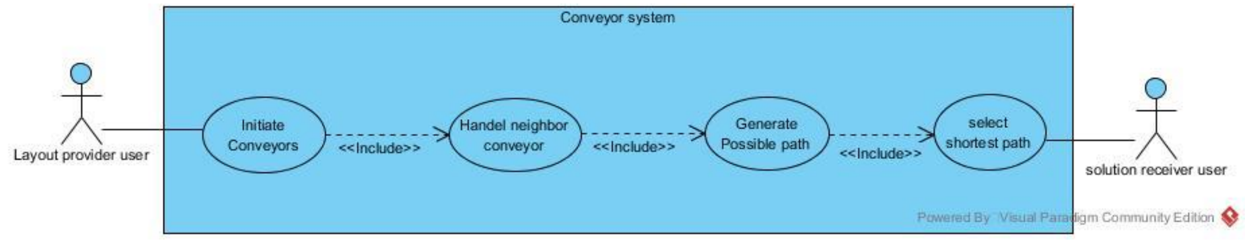


Figure 2: Use Case diagram

Step 2: Initial Agent Types Identification

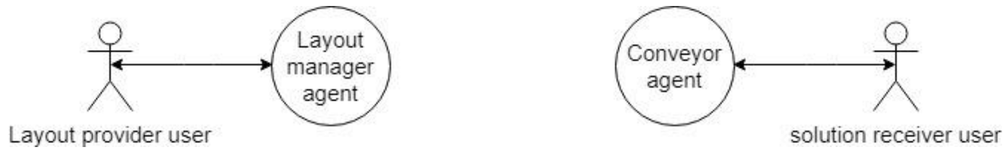


Figure 3: Agent diagram

Step 3: Responsibilities Identification

Table 1Responsibility table

Agent type	Responsibilities
Layout manager agent	• Serve requests to initiate all conveyor • Let the Layout manager agent inform Conveyor agent about the next/ neighborhood conveyor
Conveyor agent	• send requests to next/ neighborhood conveyors • generate all possible path • select the shortest path

Step 4: Acquaintances Identification

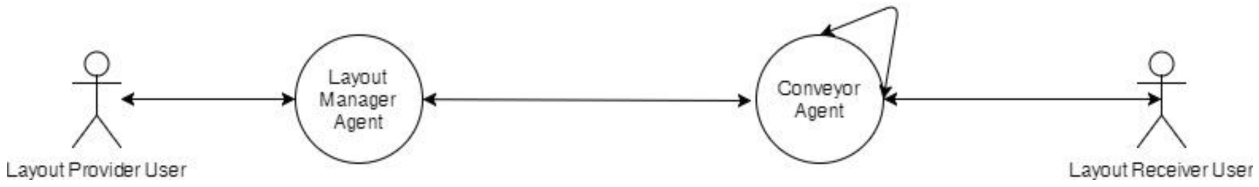


Figure 4: Agent diagram refined after Step 4

Table 2: Responsibility table updated after Step 4

Agent type	Responsibilities
Layout manager agent	• Serve requests to initiate all conveyor • Let the Layout manager agent inform Conveyor agent about the next/ neighborhood conveyor • Let the Layout provider user select the neighbor conveyor • Send the neighbor conveyors information as argument to the Conveyor agent.
Conveyor agent	• send requests to next/ neighborhood conveyors • check the destination conveyor reached or not • generate all possible path • select the shortest path

Step 5: Agent Refinement

The Layout manager agent require the neighbor conveyors of the system. Also considering Support, Discovery and Management & Monitoring, the artifacts shown in the following Figure and Table are obtained with respect to the Layout manager agent and Conveyor agent case study

Step 6: Agent Deployment Information

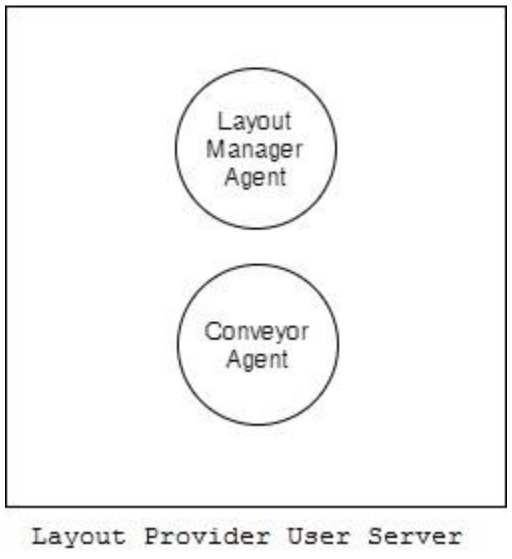


Figure 5: Agent deployment diagram

Table 3. Responsibility table updated after Step 5.

Agent type	Responsibilities
Layout manager agent	1. Create the conveyor agents 2. Serve requests to initiate all conveyor 3. Let the Layout manager agent inform Conveyor agent about the next/ neighborhood conveyor 4. Let the Layout provider user select the neighbor conveyor 5. Send the neighbor conveyors information as argument to the Conveyor agent.
Conveyor agent	1. Let the Solution receiver user select the source and destination conveyor 2. Retrieve the next conveyor from the Layout manager agent. 3. Send requests to next/ neighborhood conveyors 4. Check the destination conveyor reached or not 5. Register the via conveyor to the message 6. Check the same conveyors are not added to the message for avoiding infinite loop. 7. Brake if same via point occurred 8. Inform about the path after reaching to the destination conveyor 9. Generate all possible path 10. Compare the current possible path with the previous possible path 11. Select the shortest path

2. Design

Step 1: Agent Splitting/Merging/Renaming:

This step is considered important since it has a direct effect on overall system efficiency and complexity. Based on the given rules in the methodology, two agents are enough.

Step 2: Interactions Specification

Table 4. Interaction table for Conveyor agent

Interaction	Responsibility	IP	Role	With	When
Solution receiver user	1	FIPA Request	Initiator	Conveyor agent	At initiation
Send requests to next/ neighborhood conveyors	3,5	FIPA Request	Initiator	Conveyor agent	Destination is not same as local agent
Inform about the path after reaching to the destination conveyor	8	FIPA Inform	Initiator	Conveyor agent	Destination is same as local agent
Retrieve the shortest path	11	FIPA Inform	Initiator	Conveyor agent	Evaluating all the possible path

* Responsibility is identified in the **responsibly table** which is **produced in the analysis phase**

Step 3: Ad-Hoc Interaction Protocol Definition

No Ad-Hoc Interaction Protocol is defined. Only FIPA Protocol is adapted for interaction.

Step 8: Internal Agent Behaviors

The following Behaviors are used:

- OneShotBehaviour
- CyclicBehaviour
- WakerBehaviour

Flow chart diagram

Conveyor agents are smart entities. The algorithm logic that runs in each of them is presented in the following flow chart diagram.

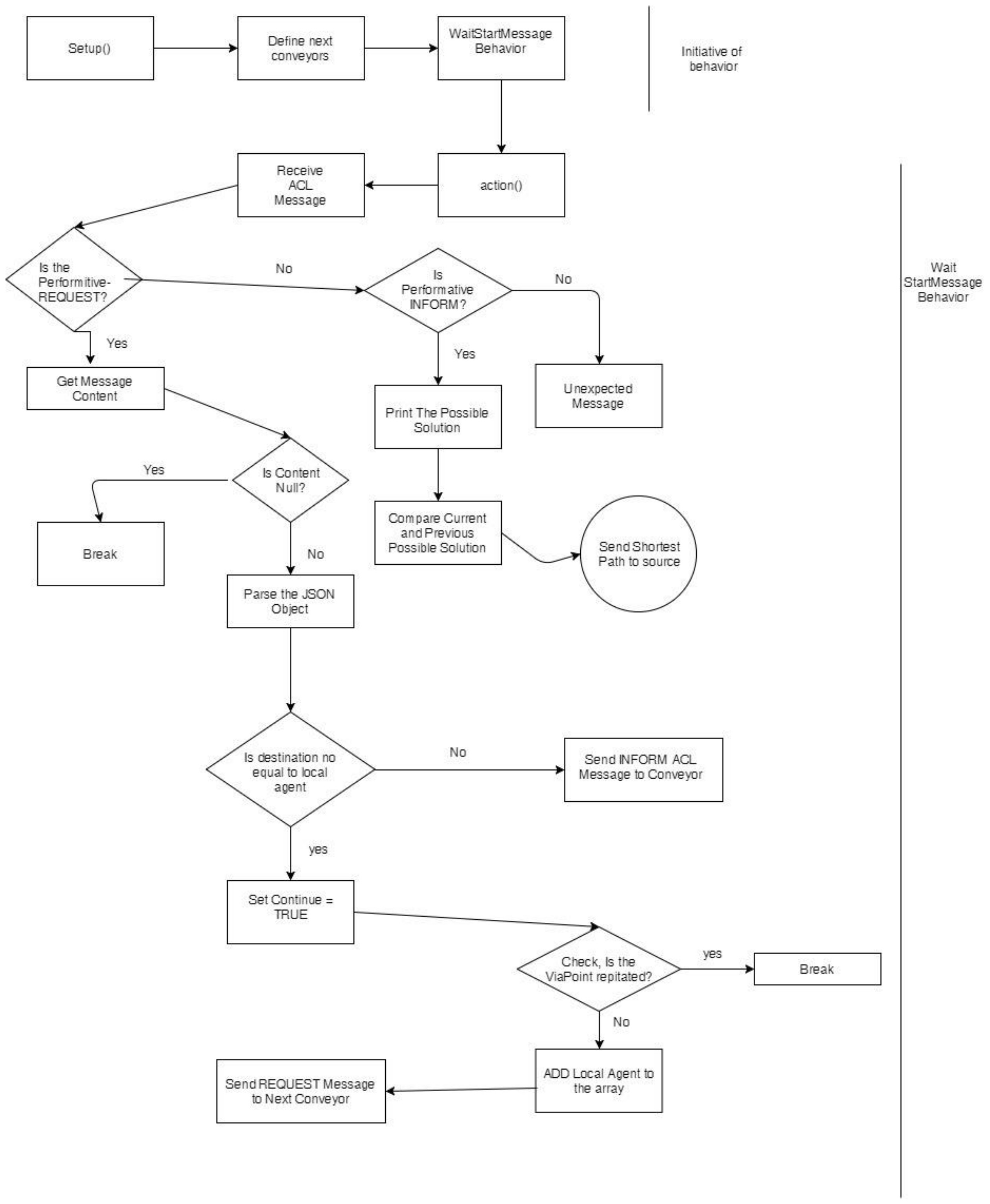

Setup()
Define next conveyors
WaitStartMessage Behavior
Initiative of behavior
action()
Receive ACL Message
Is the Performitive-REQUEST?
No
Is Performative INFORM?
No
Yes
Yes
Get Message Content
Unexpected Message
Print The Possible Solution
Is Content Null?
Yes
Break
No
Compare Current and Previous Possible Solution
Send Shortest Path to source
Parse the JSON Object
Is destination no equal to local agent
No
Send INFORM ACL Message to Conveyor
yes
Set Continue = TRUE
Check, Is the ViaPoint repitated?
yes
Break
No
ADD Local Agent to the array
Send REQUEST Message to Next Conveyor
Wait StartMessage Behavior

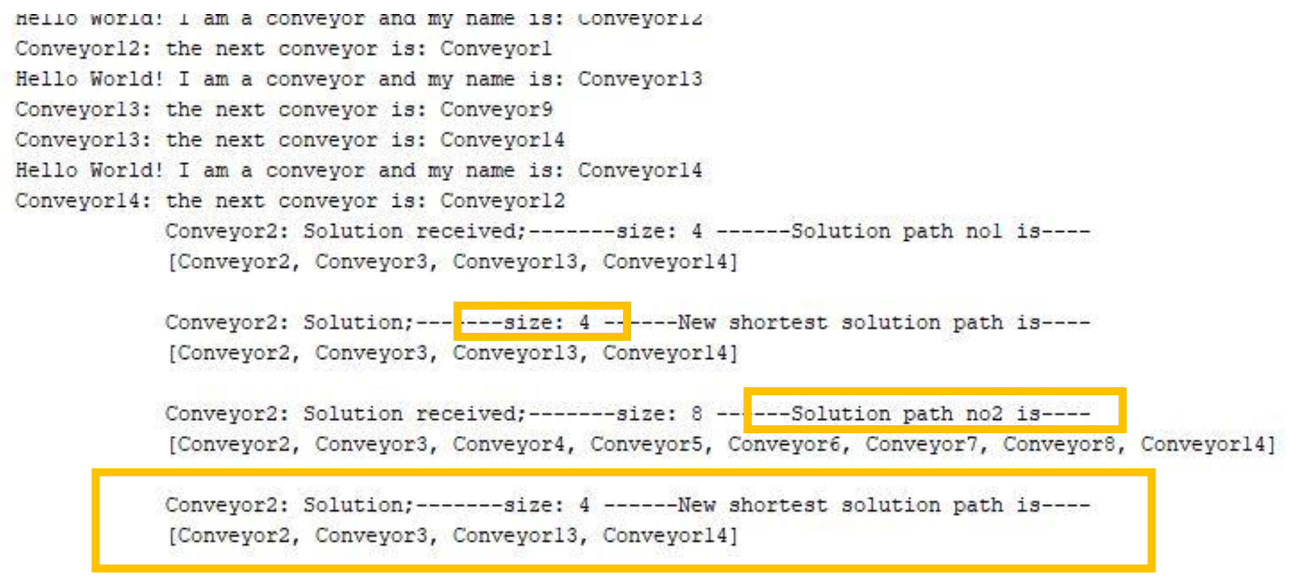
Figure 6 Conveyor agents Flow chart diagram

Feature of the developed MAS

- **Present all the possible solution**. For example: if the Dummy Agent REQUEST the following ''Massage Contain'':

  {"source":"Conveyor2","dest":"Conveyor12","midPoints":[]}

  This system shows the following result in which from Conveyor2 to Conveyor12 maximum 2 solution possible **without repeating a same via point**. It shows the size and the possible solution number.



```
Hello World! I am a conveyor and my name is: Conveyor12
Conveyor12: the next conveyor is: Conveyor1
Hello World! I am a conveyor and my name is: Conveyor13
Conveyor13: the next conveyor is: Conveyor9
Conveyor13: the next conveyor is: Conveyor14
Hello World! I am a conveyor and my name is: Conveyor14
Conveyor14: the next conveyor is: Conveyor12
          Conveyor2: Solution received;-------size: 4 ------Solution path no1 is----
          [Conveyor2, Conveyor3, Conveyor13, Conveyor14]

          Conveyor2: Solution;-------size: 4 ------New shortest solution path is----
          [Conveyor2, Conveyor3, Conveyor13, Conveyor14]

          Conveyor2: Solution received;-------size: 8 ------Solution path no2 is----
          [Conveyor2, Conveyor3, Conveyor4, Conveyor5, Conveyor6, Conveyor7, Conveyor8, Conveyor14]

          Conveyor2: Solution;-------size: 4 ------New shortest solution path is----
          [Conveyor2, Conveyor3, Conveyor13, Conveyor14]
```

Figure 7: Result

- The possible solutions and the shortest solution are sent to the Source Conveyor as INFORM message which is shown in the following picture.
- It **able to handle** requests that will actually drive your system into **"infinite" loops**.
- Robust System, it does not hang/freezes or become unresponsive and able to handle different situations.

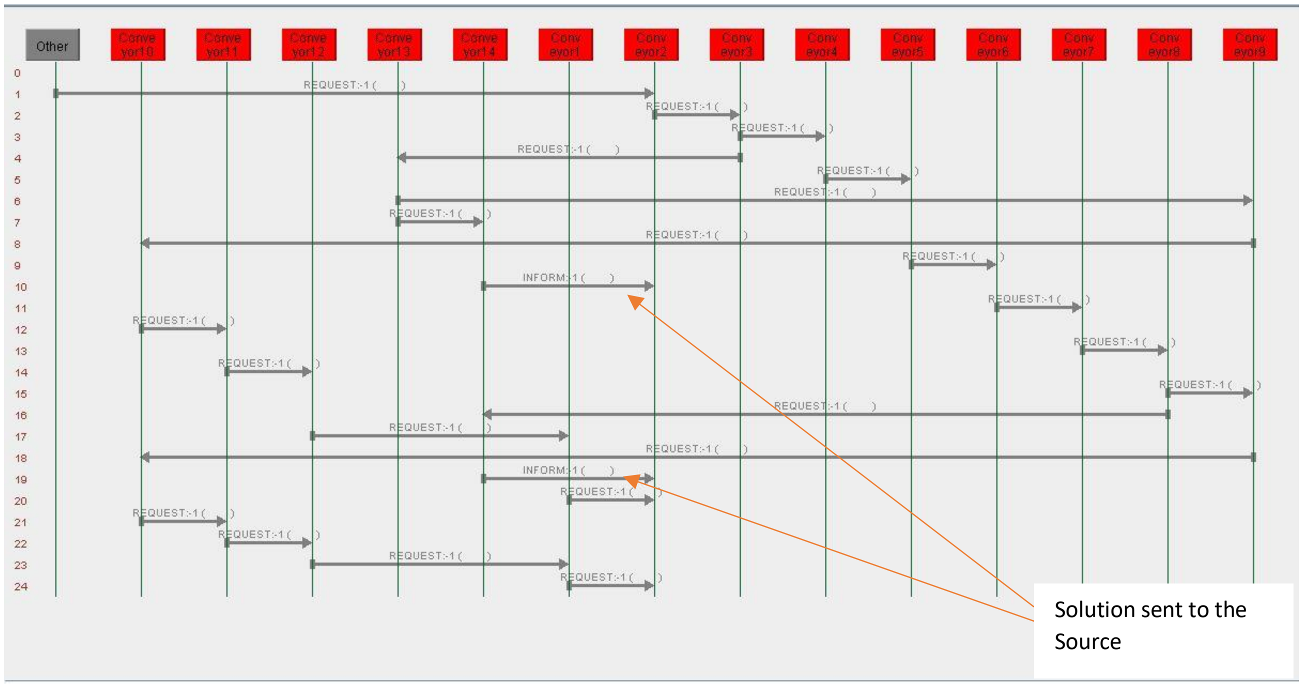


Figure 8: Interaction within Conveyors

Task 2

Question 1

JADE Behaviors:

1. Primitive Behaviors

- **Simple Behavior:** This is the basic class, that can be extended in various ways.
- **Cyclic Behavior:** As long as the agent is active, this behavior stays active. This behavior is called after every event.
  - **Ticker Behavior:** runs in a loop to execute some used defined code, It has action() and done() methods predefined to execute onTick().
- **One Shot Behavior:** This behavior executes only once.
  - **Waker Behavior:** It executes at some defined time. It has action() and done() methods predefined for execution to execute onwake().

Receiver Behavior: It executes when certain type message is received**.**

2. Composite Behaviors

- **Parallel behavior:** is defined to control children behaviors. All the children behaviors execute in parallel, and group execution termination can be defined: either when all children are executed, or when Any children are executed.
- **Sequential Behavior:** in here, children behaviors are executed sequentially rather than parallel execution and group terminates when last children behavior is executed.
- **FSM Behavior:** This behavior schedules all the sub tasks according to finite state machine. [4]

3. Delay Behavior

Delay behavior is used to introduce delays in the execution. It is started by calling onStart() method, and remaining time is calculated constantly. When the remaining time is negative, handleElapsedTimeout is called and this behavior ends. [2]

Question 2

MAS Interaction Protocols

The protocol described here are classified according to the type of agent:

Coordination Protocols

Coordination Protocol enables agents to live up to their commitment. It allows the coordination of tasks between agents that seems to provide most optimal allocation. Coordination Protocol includes acquaintance networks for distributed task allocation and Contract Network. It can be used in Business process, where different agents having same ontology, plays its role in coordination with others to perform tasks.

Cooperation Protocols

Cooperation protocol divides tasks into sub tasks and distribute them among different agents. This protocol defines the tasks for each agent and allocated resources etc. This protocol is used when tasks are quite complex, then they are divided into simpler sub tasks and allocated to different agents.

Negotiation Protocols

Negotiation protocol is used when resources are shared among agents with different goals and use of resources by one agent can comes in the way of other agent in achieving its goal. Each agents position and set criteria defines the allocation of resources etc for negotiation and decision making. The condition here is that agents must some common vocabulary/ontology to understand each other. [3]

FIPA Protocols

FIPA provides some protocols for interaction:

Basic Protocols:

FIPA protocols use ACL, and they are simple request protocols that an agent uses to ask another agent to perform a task or request information etc. When one agent makes a request, the receiver agent indicates whether it'll accept or reject the agent. If agent accepts the request, it also indicates the application when It performs the ask requested.

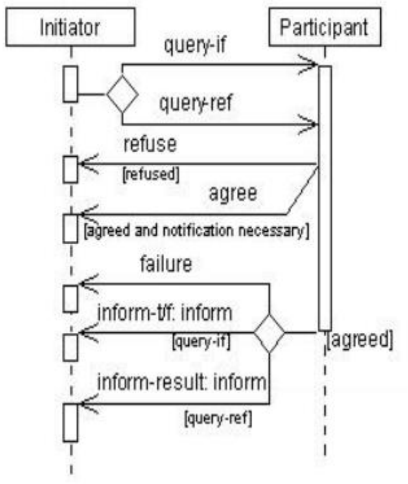


Figure 9: Conditional Query Protocol FIPA

The picture shows the conditional query language protocol, this allows an agent to make some request to other agent when a certain condition is met. When the condition is met, the other agent accepts the request and informs the back that action has been performed.

Network protocols contractual FIPA:

This protocol allows an agent (manager) to make a bid for a task, different agents offer their services, and the based on the offers, manager selects an agent to do the job. The agent who gets the job informs back when job is completed.

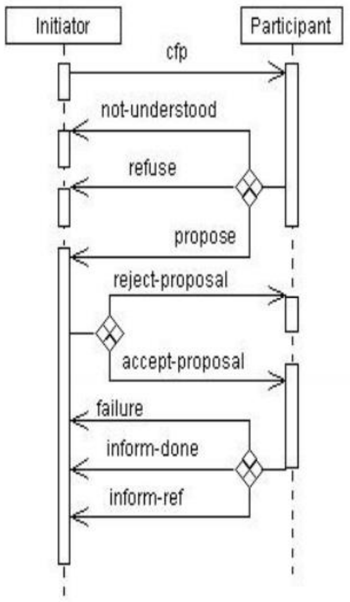


Figure 10: Network protocols contractual FIPA

Protocols FIPA auction:

There are 2 versions of this protocol:

1. Protocols FIPA Dutch Auction

In this protocol, seller agent set a price that is way higher than the actual price and price is reduced until someone accept the price.

2. Protocols FIPA English Auction

In this auction, price is set lower than the actual price and interested buyers keep on raising the offered price until no one wants raise the price.

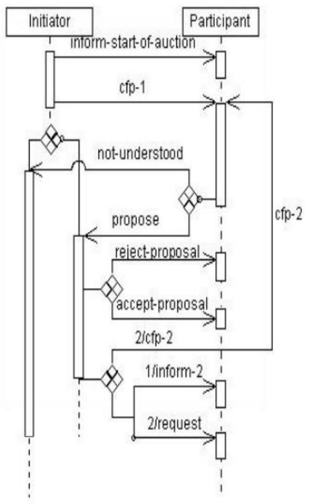


Figure 11: Protocols FIPA English Auction

These Auction Protocol are widely used in electronic commerce.

Question 3

Research Paper Summary:

Abstract

This document contains the review of paper "Applications of Multi-Agent Systems in Smart Grids: A Survey" [1] written by Ghezlane Halhoul Merabet, Mohammed Essaaidi, Hanaa Talei, Mohamed Riduan Abid, Nacer Khalil, Mohcine Madkour, Driss Benhaddou.

Summary:

Smart grids are considered as new power generation systems, and they are combined with the artificial intelligence, Information technology and distributed systems for real-time energy consumption monitoring. For the management of distribution systems, MAS is used. Power consumption is being shifted from large centralized systems to the small de centralized and distributed systems. The future of SG will be such that mesh networked microgrids will be providing energy to the consumers. But for this to happen, IT systems will be needed to pass information and to make autonomous actions/decisions for the generation, distribution of power at subsystems. In SG distributed system, MAS is useful because they have local decision-making powers, and only useful information is exchanged between the agents, So, the amount of data needed to be transmitted is reduced and thus the cost of information systems. Since, MAS systems are distributed systems, and failure of one systems doesn't effect the whole process, so, in here, failure of one power generation system in SG will not effect the power distribution and agents will adapt to the new change. MAS are quite flexible, they can add or remove resources and capabilities, so they demonstrate plug and play concept for SG. In SG, it is important that to predict the power consumption and generation patterns by different consumers to work autonomously accordingly, MAS having the data processing and decision making capabilities at agent levels make it quite easy to implement such systems using machine learning and problem solving algorithms. The MAS approach is applied in 2 steps: development of a system for management of energy in turbine and second is to develop a Demand/Response (DR) system to shift local loads. If the local controllers are smart and able to communicate with each other, fully autonomous MAS can be achieved. Machine to Machine communication is used for the transfer of real time production and consumption of data. ZEUS is a MAS Platform for multi agent distributed systems. ZEUS deals with Agents, their goals, their tasks and their facts. But it supports only agent system. While AgentBuilder is a commercial software for intelligent agent system designing. It is used for developing ontology and inter agent communication. JADE is multi agent platform and it is designed for interoperable MAS. FIPA enables the communication of JADE agents with non-JADE agents.

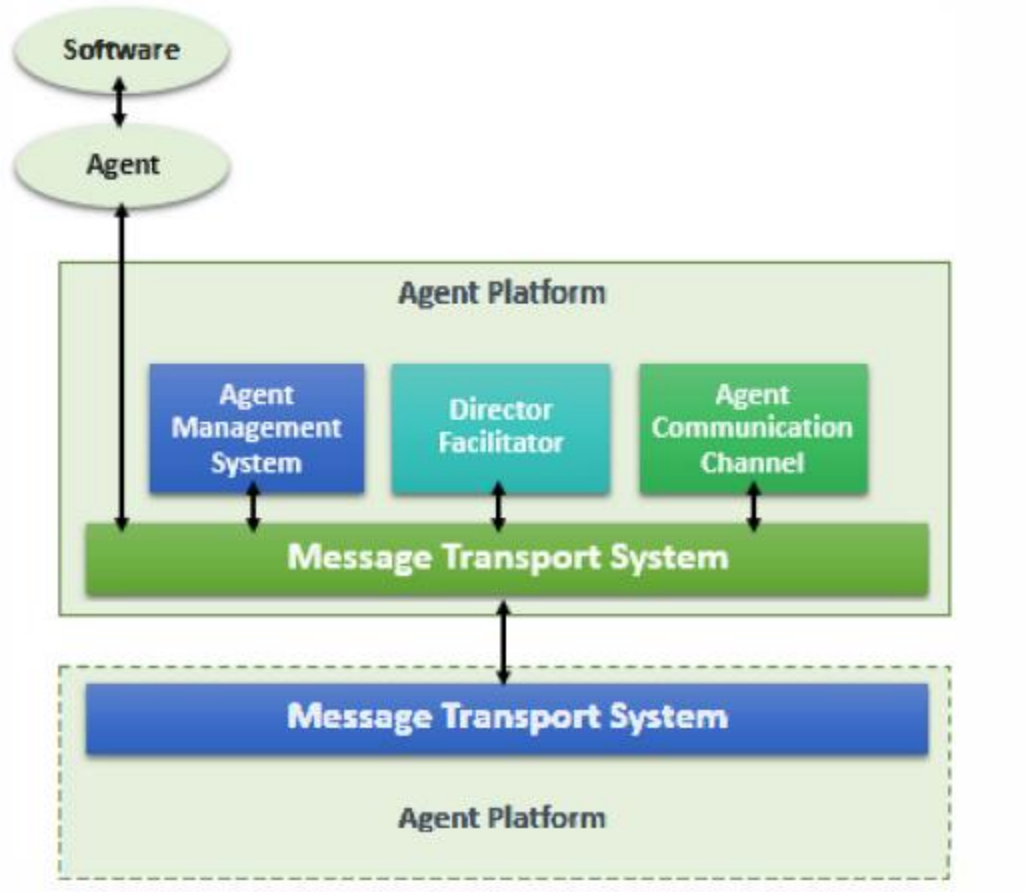


Figure 12 JADE Architecture

JADE is well suited to implement mobile agents and JADE environment is such that it provides satisfactory performance. JADE provides: Independent performances, better tolerance and reduced network load. MADKit (Multi-agent development kit) is most flexible platform and It can adapt to different models. Like JADE, it is also written in JAVA. In MADKit, different agents are placed in different groups and each agent plays its role exactly like object-oriented programming. MADKit is based on AGR i.e Agent-Group-Role.

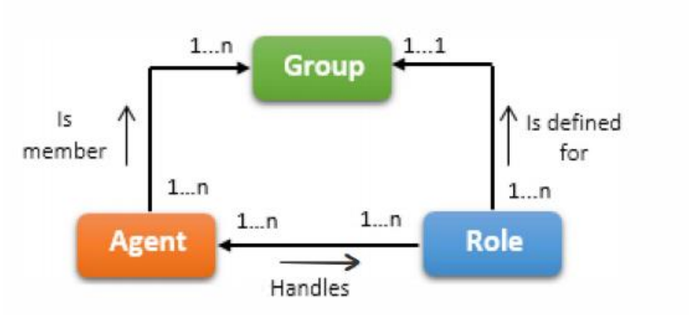


Figure 13MADKit AGR Model

In conclusion, there are different possible platforms to implement SG with MSA. But this implementation will provide more efficient energy, distributed and autonomous energy systems.

Reference:

1. Halhoul Merabet, G., Essaaidi, M., Talei, H., Riduan Abid, M., Madkour, M., Benhaddou, D. and Khalil, N. (2018). Applications of Multi-Agent Systems in Smart Grids: A survey - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6911384 [Accessed 27 May 2018].
2. Iro.umontreal.ca. (2018). Jade Behaviours. [online] Available at: https://www.iro.umontreal.ca/~vaucher/Agents/Jade/primer6.html#6.1 [Accessed 28 May 2018].
3. Marzougui, B. and Barkaoui, K. (2018). [online] Available at: https://www.researchgate.net/publication/270553738_Interaction_Protocols_in_Multi-Agent_Systems_based_on_Agent_Petri_Nets_Model?enrichId=rgreq-946d71cccda4b435f8e51cc4ee9adefe-XXX&enrichSource=Y292ZXJQYWdlOzI3MDU1MzczODtBUzozNzI0NTE3NjM2MDU1MzdAMTQ2NTgxMDgyOTAwMg%3D%3D&el=1_x_2&_esc=publicationCoverPdf [Accessed 28 May 2018].
4. Jade.tilab.com. (2018). [online] Available at: http://jade.tilab.com/doc/tutorials/JADE_methodology_website_version.pdf [Accessed 3 Jun. 2018].