The Pennsylvania State University

The Graduate School

The School of Science, Engineering and Technology

MULTI AGENT SYSTEM-BASED SIMULATION OF A LABORATORY-SCALE MICROGRID

A Thesis in

Electrical Engineering

by

Le Chen

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2014

The thesis of Le Chen was reviewed and approved* by the following:

Peter Idowu
Professor of Electrical Engineering
Assistant Dean for Graduate Studies
Thesis Adviser

Aldo Morales
Professor of Electrical Engineering
Co-Director, Center for Signal Integrity

Seth Wolpert
Associate Professor of Electrical Engineering

Mohammad Tofighi
Associate Professor of Electrical Engineering

Jeremy Blum
Associate Professor of Computer Science

*Signatures are on file in the Graduate School.

ABSTRACT

The Multi-Agent-System (MAS) technology has many desirable attributes such as autonomy, sociality, reactivity and pro-activity. It is widely accepted as the technology platform for implementing effective and efficient management and automation processes within a microgrid environment. This paper proposes an implementation of a microgrid simulation utilizing Matlab and a MAS software program. The MAS software is implemented with the aid of the Java Agent Development Framework (JADE) middleware platform. The three intelligent agents are: Controller agent, Distributed Energy Resource (DER) agent and Load agent. The Controller agent monitors network processes, performs critical control task such as network reconfiguration and it is also capable of detecting network anomaly. The DER agent stores the associated energy resources information, monitors and controls the DER power levels. Finally, the Load agent stores information about the users and loads such as power consumption and the priority status of the load. Both DER and Load agents are able to interact and respond to Controller agent's command for connecting / disconnecting from the power network. This simulation will demonstrate the benefits of employing a standard MAS environment that could serve as a platform for studying real-time microgrid's communication, monitor and control technologies.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Meaning	Page
ACL	agent communication language	6
AMI	advanced metering infrastructure	8
AMS	agent management service	11
ANN	artificial neural networks	39
CERTS	consortium for electric reliability technology solutions	3
CFP	call for proposal	23
CIM	common information model	22
DER	distributed energy resource	2
DF	directory facilitator	11
DG	distributed generation	1
DLC	direct load control	1
EV	electrical vehicle	2
FIPA	foundation for intelligent physical agents	10
IEC	international electrotechnical commission	22
JADE	Java agent development framework	6
JVM	Java virtual machine	38
KQML	knowledge query and manipulation language	10
MACSimJX	multi-agent control for Simulink program	6
MAS	multi-agent system	5
NTUA	National Technical University of Athens	5
OO	object-oriented	22
PHEV	plug-in hybrid electric vehicle	2
PSH	Penn State Harrisburg	1
RF	radio frequency	2

Acknowledgement

Foremost, I would like to express my sincere gratitude to my advisor Dr. Peter Idowu for his continuous support and his guidance throughout the thesis project.

Besides my advisor, I would like to thank Dr. Seth Wolpert, Dr. Mohammad Tofighi and Dr. Jeremy Blum for providing their time to review the thesis report and providing valuable feedback. Furthermore, I am grateful to Dr. Jeremy Blum for providing valuable insights during the software development part of the project.

I thank my fellow graduate classmates in Penn State Harrisburg for the stimulating discussions and last but not the least, I would like to thank my family: my parents and sister for their spiritual support throughout the project.

Chapter 1. INTRODUCTION

This chapter will introduce the premise surrounding the idea of smart grid. Then the concept of microgrid and its technology will be introduced along with an overview of the Pennsylvania State University - Harrisburg (PSH) microgrid implementation. Finally, the goals and objectives of the thesis project within the framework of the PSH microgrid will be described.

Smartgrid

Smart-grid is a “buzz word” that encompasses many areas of disciplines beyond the traditional power grid and its systems. All things considered, smart grid is socially transformational [1]. The “Smart grid” refers to a class of technology that utilizes computer-based remote control and automation. These systems are made possible by two-way communication technology and computer processing that has been used for decades in other industries (*U.S. Department of Energy)*

While much of the sophistication had already been implemented at the transmission level of electricity, many areas of development are still needed; especially at the distribution level. It is believed that smart grid technologies could help to provide the intelligence and sophistication needed for a modern power system. Many motivations are driving the development and deployment of the advanced power grid technologies such as rising cost of energy, new market model due to energy deregulation, the availability of renewable technology and an overall aging infrastructure that could affect the quality and reliability of electricity and pose security issues. The stakeholders in the smart grid initiatives are the consumers, private/commercial sector, government, research and academic institutions and business/financial sector.

While smart grid is sometimes criticized for its shortcomings, such as in personal privacy and potential adverse health effect caused by its wireless system, it remains largely praised for the potential positive outcomes from its deployment. The inception of smart grid technologies is accompanied by numerous merits in areas of technical efficiency, cost, energy security and environmental preservation. Among many “smart” attributes of a smart grid, being able to discern the critical application from the common ones is an invaluable capability. Using the data collected through smart devices within its Advanced Metering Infrastructure (AMI), the operator could potentially allocate more of the grid’s resources to ensure the power quality is conforming to certain mission-critical applications [1, 2]. For example, in a case of power instability due to heavy loading, the utility may opt to use Direct Load Control (DLC) in order to switch off non-essential loads - to provide momentary power compensation. The initial capital investment for smart grid might be exorbitant for many, but a grid with

smart technology could with time lower the cost of operation, maintenance and expansion. For instance, higher Distributed Generation (DG) penetration could potentially curb the need to build new power transmission infrastructure, which will reduce the cost of grid expansion while increasing the efficiency of power utilization.

Few hurdles still lie ahead. As of 2009, only 15 states have the interconnection standards that are considered favorable for distributed generation [1]. Also, concrete short-term financial incentives for end-users and power producers need to be defined in order to provide the impetus for further proliferation of DGs. Technological challenges in areas of energy storage systems also hinder the deployment of smart grid technology as well as for the integration of different renewable resources and the interoperability of smart grid standards. Furthermore it is difficult to establish standards when the technology is evolving; smart grid is a field where computer science, communications, information technology, economics and other disciplines congregate and these constituents are also under constant change [1, 2].

Microgrid

A microgrid could be thought as a miniature smart grid. More specifically, a microgrid is defined as an electrical network, composed of interconnected loads and Distributed Energy Resources (DER) that could operate in both grid-connected mode and in the autonomous island mode [3]. A microgrid typically incorporates advance monitoring, control and communication systems. The deployment of smart meter systems is an example of advanced monitor systems that provides a cost effective solution and allows a two-way information flow - through a variety of communication technologies such as Power Line Carrier (PLC) or Radio Frequency (RF) [4]. Advanced control system research is also very active in applications where the microgrid has direct access to loads within a house for example [5, 6].

Renewable energy technology has taken a great leap forward. For example, there is an increasing number of Electric Vehicles (EVs) and Plug-in Hybrid Electric Vehicles (PHEVs) that are online and participating in the energy exchange. The same increase in Distributed Generation (DG) adoption trend is seen with wind turbine, solar and other renewable energy sources. The increase in localized power production will inevitably entail the need to develop storage solutions in order to capture the energy - for either local use or pushing it back to the utility. Furthermore with the increasing number of DGs, there is a need to maintain normal operation at the distribution level [7, 8]. This is where the microgrid concept is introduced. A microgrid is essentially a miniaturized version of the smart grid and both share common advanced technologies. A microgrid has a common coupling point to the utility which allows it to operate in a normal

grid-connected mode or in islanding mode. Aside from lowering the cost of electricity, a well-structured power network such as the microgrid could provide the following added-benefits:

1. Provide power during emergency situations
2. Shave peak-load
3. Increase overall grid stability and reliability
4. Provide a market for producers and consumers to interact
5. Provide engineering opportunities to advance science and technology

There are many microgrid implementations in both commercial and academic settings. These test-beds host many advanced communication, monitor and control systems. A simple microgrid layout is shown in Figure 1. At its essence, it is composed of an AC network and a DC network with loads and generators in their respective network. These two networks are interfaced through a bi-directional inverter.

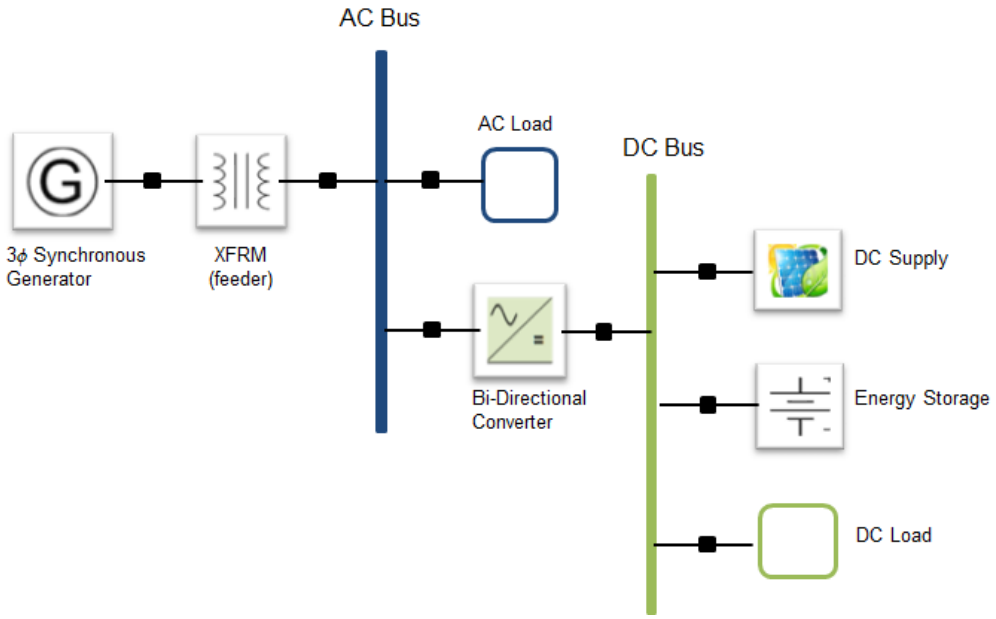

Figure 1, Simplified topology of a microgrid network

Table 1 highlights few examples of microgrid test-beds.

Microgrid	Description
CERTS (Columbus, Ohio)	The Consortium for Electric Reliability Technology Solutions (CERTS) develop and disseminate electric reliability technology solutions in order to protect and enhance the reliability of the

	U.S. electric power system under the emerging competitive electricity market structure. Areas of research include Real-Time Grid Reliability Management Reliability & Markets Distributed Energy Resources Integration Load as a Resource Reliability Technology Issues & Needs Assessment [9].
University of California Irvine	Testing how microgrids operate internally as well as how they interface with the rest of the future smart grid. It is a test bed for different technologies through the development of the UCI Microgrid model, deployment of advanced metering and various pilot projects. The campus provides an attractive platform to support a flexible and robust platform for the deployment and evaluation of the various technologies and circuit configurations emerging in the microgrid future [10].
Jeju island (Korea)	World's largest Smart Grid community (6000 households) that provides a testing ground for advanced Smart Grid technologies, R&D, development of business models and verification of different power market models. More specifically the five areas of research are: smart power grid, smart place, smart transportation, smart renewable and smart electricity service [11].

Table 1, Example of microgrid implementations

The microgrid that is currently under development at the Pennsylvania State University - Harrisburg (PSH) will also showcase advanced hardware and software systems. The test-bed will also be used to validate existing technologies and concepts in areas of communications, monitor and control. In addition, it will be served as a platform for developing advanced concepts and technologies in areas of data mining, economic dispatch, optimal control, wireless communication and other emerging smart grid concepts.

Albeit smaller in scale and capacity in comparison to the microgrid initiatives described in Table 1, the PSH microgrid test-bed will serve as a platform to study technologies which might not be readily applicable in an industrial or commercial setting, but are nevertheless paradigm-shifting. The microgrid under development at PSH is shown in Figure 2.

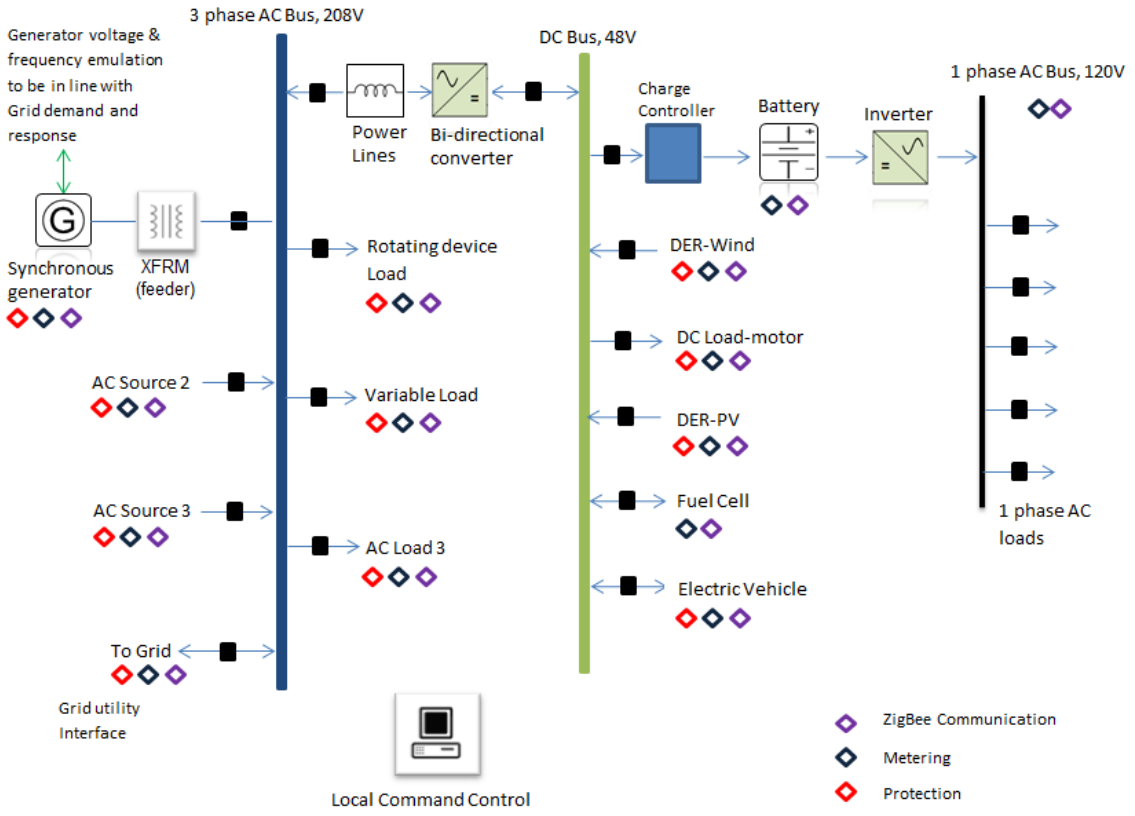

Figure 2, Proposed PSH microgrid test-bed

Various approaches have so far been proposed for modern power system control. For example, during an event of power interruption due to a fault, power restoration is required where an optimal target configuration is determined. Currently, many methods in power restoration are employed. They could be roughly classified into four categories: heuristics, expert systems, mathematical programming and soft computing. Heuristics and expert systems have been used in industries extensively, but they both have their own deficiency with respect to the optimality of solutions. On the other hand, mathematical programming could produce the optimal solution after the formulation, but it requires engineering judgment in formulating restoration problems. Furthermore, the long execution time required for mathematical programming makes it impractical considering the time constraints on site. At last, soft computing methods are easy to implement but they cannot obtain the optimal solutions in the true sense and they also require long computation time [12].

The microgrid has a distributed architectural which provides a platform where grid entities could participate in energy and information exchange. Many of the advanced interactions between these entities, such as determining the price of energy, could be hard to implement. Also the dilemma of software inter-operability is a concern. Ideally, within the microgrid, devices such as generators

or loads should operate seamlessly regardless of the equipment manufacturer. It is only with a flexible and extensible and software approach that we could ultimately provide a rich platform where advanced microgrid processes and device interactions could take place and where the plug-and-play capability could facilitate microgrid integration with minimum software development.

Goals and objectives

Some microgrid implementations are currently experimenting with Multi-Agent System (MAS) technology – an advanced software system. One such implementation is by the National Technical University of Athens (NTUA) in cooperation with ANCO S.A. on the pilot microgrid of Kythnos island, Greece where the microgrid's upstream and downstream monitor and control are managed by an agent system [5]. More specifically, the agents provide power management though a Direct Load Control (DLC) process. For instance, in case of a power emergency, a higher-level agent would notify a lower-level local agent to shed off non-essential loads such as a water pump.

In this thesis, it will be shown that the MAS could be a viable solution for the shortcomings in modern microgrid control, power system extensibility and for the microgrid's lack of rich sets of interaction.

The long term objective of PSH power engineering group is to implement a physical microgrid in order to provide test-bed to study emerging smartgrid technologies, such as MAS. Prior to deploying the advanced technology, an assessment of the approach must be performed. In the framework of this project, a microgrid simulation will be performed. More specifically, a co-simulation platform involving an intelligent agent approach is explored. This simulation will hopefully pave the way for building the hardware-based microgrid with smart agent capabilities.

Previous work [13, 14] on power system co-simulation had shown that MAS could be used to perform a simple load balancing process in an event of microgrid islanding. This project will illustrate an advanced co-simulation between a microgrid (in MATLAB) and a novel communication, monitor and control system implemented using a MAS software (JADE). A gateway interface program (MACSimJX) will manage the data flow between the microgrid and the MAS program. In the scope of this project, the primary roles of the MAS will be to perform a network reconfiguration tasks by balancing the loads and dispatching the DER in an event of microgrid islanding.

Details about the microgrid program, MAS program and the interface program will be discussed in following chapters. It could be observed that this project brings several improvements over the previous co-simulation work in terms of the

complexity of the microgrid simulation as seen in [6, 13, 15] and most importantly by employing a different agent architecture design as seen in [6, 13, 14, 16]. Table 2 summarizes the key differences.

Previous work	Focus of research
A single generator with different power levels. One single switchable load [6, 13, 15].	A realistic microgrid with many switchable generators and loads.
The DER and Load agents have a managerial role. Both agents have a priori power production / consumption information of each DER and load [6, 13, 14, 16].	DER and Loads agents are instances of a class definition hence make use of data encapsulation for sending agent data. Every agent created from the “agent template” is independent of each other and is able to perform local tasks (i.e. acquire real-time data).
Use a simple "*Inform*" Agent Communication Language (ACL) communication [6, 13, 14, 16].	Make use of "*call-for-proposal / propose / accept / inform*" ACL communication. This lays the ground work for a Contract Net Protocol that could be used on the PSH microgrid

Table 2, Co-simulation comparison

The proposed architecture makes use of object-oriented-programming to allow software code reuse and hence better exemplifies the concept of distributed networks by embedding the agents into the different microgrid entities or equipment such as DER and load [17, 18]. The modularity of the proposed MAS software architecture could facilitate microgrid expansion by cutting down on individual device software development. Furthermore, as it will be shown, the proposed software architecture along with the open MAS platform could promote the interoperability between different electric equipment manufacturers and allow operation on different computing platforms.

These merits are very attractive for the microgrid implementation at PSH. This co-simulation should provide a good assessment of the agent technology. The agent concept (Controller / DER / Load agents) as well as the software developed for the co-simulation could be readily ported onto the real-time microgrid network composed of many distributed DER and loads.

Chapter 2. CONCEPT AND DESIGN

In this chapter, the different software constituting the co-simulation platform will be described. First, the microgrid implemented in MATLAB will be explained. Then the MAS concept will be introduced along with an agent framework called JADE. At last, MACSimJX - the software which links the simulations running in MATLAB and in JADE will be described.

Microgrid simulation

In this thesis, a simulation of a microgrid network will be analyzed. Using an intelligent agent approach, it will be shown that a microgrid operation such as islanding (when microgrid disconnects from the upstream utility) could be achieved using an open architecture agent-based software. It is believed that this distributed control software used in the simulation could serve as a worthy candidate for the real-time microgrid control system implementation.

Matlab's Simulink with the SimPowerSystem extension is used to build the microgrid. For the purpose of this study, the microgrid's main transformer is rated at 10 KVA with 7 kV primary and 208 V secondary supplying power to a three-phase 208V AC network. The AC network is comprised of four 3-phase loads (labeled as "Load X" in Figure 3) and four 3-phase synchronous generators (labeled as "DER X" in Figure 3). The microgrid feeder is simulated by 100 MVA source. The microgrid network under simulation, shown in Figure 3, is a close representation (in terms of setup and capacity) to the one that is currently under development at PSH.

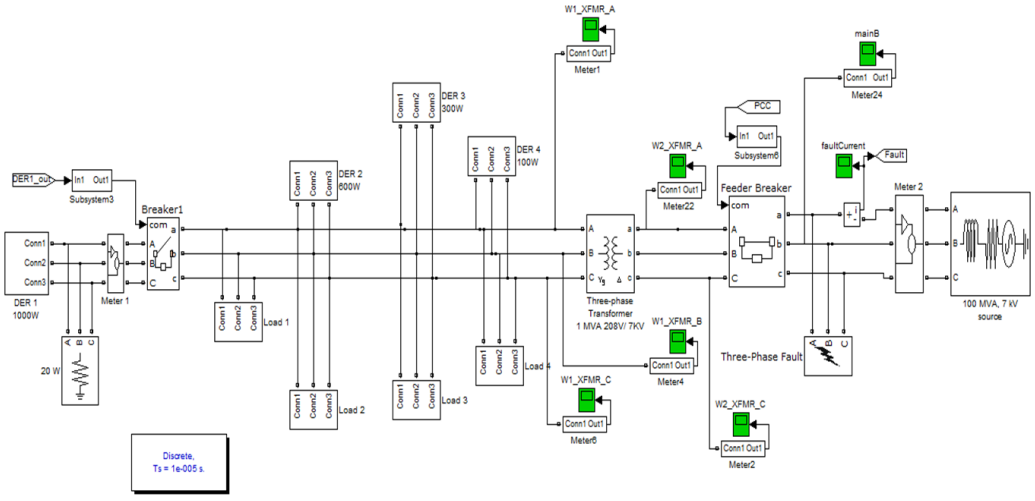

Figure 3, Co-simulation microgrid under study

Shown in Table 3 are the ratings for the DER and loads. Initial simulation of the microgrid was performed in order to verify DER/load switching, voltage transient, fault generation, etc. The discrete-time ode45 solver (for accurate computation for non-stiff problems) was used with a 10 micro-seconds step, which translates to 100 kHz sampling time. As in most microgrid implementation, the total power capacity could be insufficient if all loads were to be supplied (i.e. 1700 W produced versus 1900 W consumed).

Component / Rating	Power	Voltage (V)
Feeder	100 MVA	7000
Transformer	10 kVA	7000/208
DER1	700 W	208
DER2	600 W	208
DER3	300 W	208
DER4	100 W	208
Load1	800 W	208
Load2	600 W	208
Load3	300 W	208
Load4	200 W	208

Table 3, Component rating

Multi-Agent System (MAS) concept and design

Both smartgrid and microgrid are distributed networks which incorporate many advanced communication, monitor and control technologies. It has been suggested that intelligent agent architecture could be adopted for such distributed networks. The Multi-Agent System (MAS) is inherently efficient in solving problems in a distributed environment [19]. It could be utilized to provide a powerful platform to develop and test innovative communication, monitor and control technologies in order to further increase the efficiency and effectiveness of the microgrid.

MAS has been previously used in fields of computer science and artificial intelligence [12]. Examples of applications are planning, process control and communication network configurations [3, 12]. With the advent of smart grid technologies, MAS has been getting more attention in recent years from the power engineering world. In power engineering applications, MAS have a tendency to be exploited in two ways: as an approach to building flexible and

extensible hardware/software systems or as a modeling tool to describe complex systems and relationships by encapsulating data [1].

In a nutshell, an agent system is a software program which contains an agglomeration of different disparate agents, working in collaboration, to pursue assigned tasks and to achieve the overall goal of the system [15]. Its advantages include autonomy, sociality (ability to communicate with other agents), reactivity and pro-activity [15]. The MAS is envisioned as the future of an automated power monitor and power control system such as the SCADA [13]. For example, MAS could be used instead of a traditional zonal protection scheme in regards to fault handling [12, 13]. Furthermore it is believed that the MAS could provide added-intelligence to the PSH microgrid operations and processes in areas of distributed control and advanced ability to engage in complex inter-agent interaction.

There might be a lack of clear demarcation between MAS and conventional distributed software architecture with agent-like attributes. The fact is that many of these software architectures do not have the ability which provides inter-platform compatibility. In order to fully benefit from MAS technology, inter-platform communication must be agreed-upon. Early MAS used proprietary communication languages. Other systems have also used blackboard system-type approaches to enable communication between agents. One of the first Agent Communication Languages (ACL) to be used by different researchers across different fields was the Knowledge Query and Manipulation Language (KQML). In recent years, KQML has been superseded by FIPA-ACL. A FIPA-ACL message specifies 22 performative acts or speech acts in its message. By classifying the message using a performative, FIPA-ACL ensures that recipients will understand the meaning of a message in the same way as the sender, removing any ambiguity about the message's content [20].

Therefore without standardization, interaction between MAS programs developed by different parties could not be guaranteed. In light of increasing use of MAS in the power system world, standardization is very important. Open agent architecture does not place any restrictions on the programming language or origin of agents joining the system. This type of architecture is achieved through adherence to messaging standards: An example of a set of standards is that defined by the Foundation for Intelligent Physical Agents (FIPA) – a standards IEEE committee as of 8 June 2005. Amongst few existing open standard MAS platforms such as ZEUS and JACK, Java Agent DEvelopment Framework (JADE) is being explored in the thesis. JADE is chosen for this project simply because of the large availability of technical literature and for its overall popularity within the agent development community.

The MAS is implemented using JADE, a Java framework for developing FIPA-compliant agent applications. This middleware platform allows the developer to focus on the logical aspect of the system while it controls the communications

between different hosts and the message exchange between agents [21]. Using JADE agent environment, different computer programs (or agents) performing distinct sets of tasks could send, receive messages and interact with each other. The JADE platform provides the multi-threading capability that is needed in a multi-agent environment [22]. JADE also includes a runtime environment where the JADE agents can live, a library of classes that could be used to create new agents and a graphical tool that could be used to administer and monitor agent activities.

The JADE agent environment is composed of a series of platforms and containers. A container could be understood as a computation entity on which the Java Virtual Machine is running. The agents, shown as A1-A5 in Figure 4, reside inside a container and there could be many containers but only one main container on each platform. In the case where another main container is created somewhere in the network, this container will constitute a new platform. An agent is inherently distributable and has no fixed ties to its environment. That said, the instances of the platform running on separate machines seamlessly connect and appear as a single instance [23]. The architecture is illustrated in Figure 4:

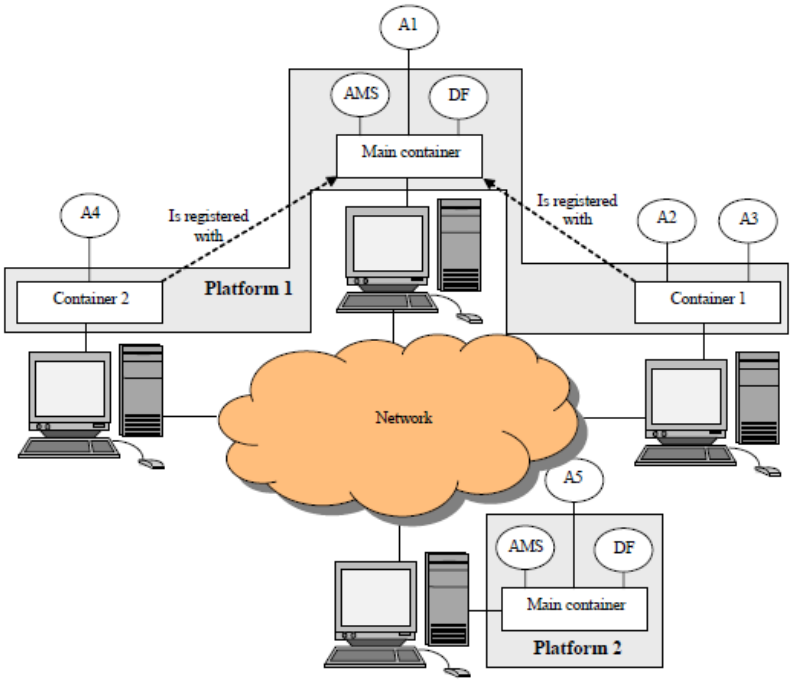

Figure 4, JADE architecture

As seen in Figure 5, each agent platform includes two utility agents: the agent management service (AMS) agent and the directory facilitator (DF) agent. The former is compulsory while the latter is optional. The AMS acts as white pages, maintaining a directory of agents registered with the MAS platform. The DF acts as yellow pages, maintaining a directory of agents and the services they can

offer other agents [20]. As it will be illustrated later in the project, an agent can use the DF to search for other agents that can provide the desired services in order to aid it in fulfilling its own particular goals.

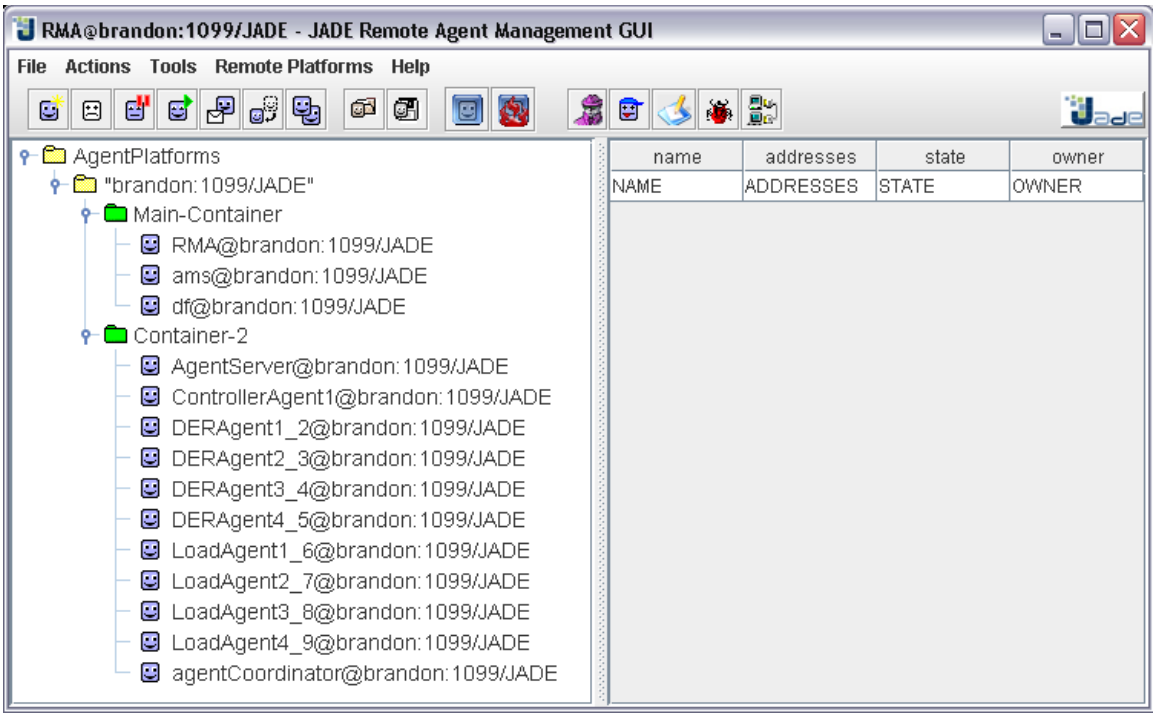

Figure 5, JADE user interface

The scope of the thesis is to illustrate the islanding process using a MAS approach. Three agents are created for this purpose and they are: **Controller agent,** the **Distributed Energy Resource (DER) agent** and the **Load agent.** These agents, which currently constitute the MAS, could be considered as the "backbone agents" of the microgrid MAS. In other words, they are responsible for the essential network tasks and once they have been established, many other types of agents could be easily created and added to the MAS environment.

The Controller agent monitors the network parameters such as voltage and frequency [6, 13, 14, 15]. It is also responsible, for instance for making control decisions in an event when a hazardous condition is detected. The Controller agent performs critical decisions based on the inputs provided by the DER agents and the Load agents. Some examples of critical decisions are: load shedding, islanding and fault handling.

The DER agent is a program which stores the associated energy resource information. It could also monitor and control the DER power levels [6, 13, 14, 15]. The DER agent responds accordingly to the Controller agent's signal to

connect/disconnect an energy resource.

Finally, the Load agent stores information about the users and loads such as power consumption and the priority status of the load. The Load agent is also able to interact and respond to the Controller agent's command for connecting/disconnecting any load from the power network [6, 13, 14, 15].

For the islanding study, a total of nine agents are created; four DER agents, four Load agents and the Controller agent. The idea is to generate a sensible number of random combinations of DER and loads to test the load shedding and DER dispatch algorithms described later in Chapter 3. Theoretically, all the agents could either run on the same PC or across many PCs located on the same or different network where the JADE platform is pre-installed on each PC. Each agent runs on a single Java thread. Each Java thread has many behaviors that could be concurrently executed. The JADE agent has the following structure shown in Figure 6.

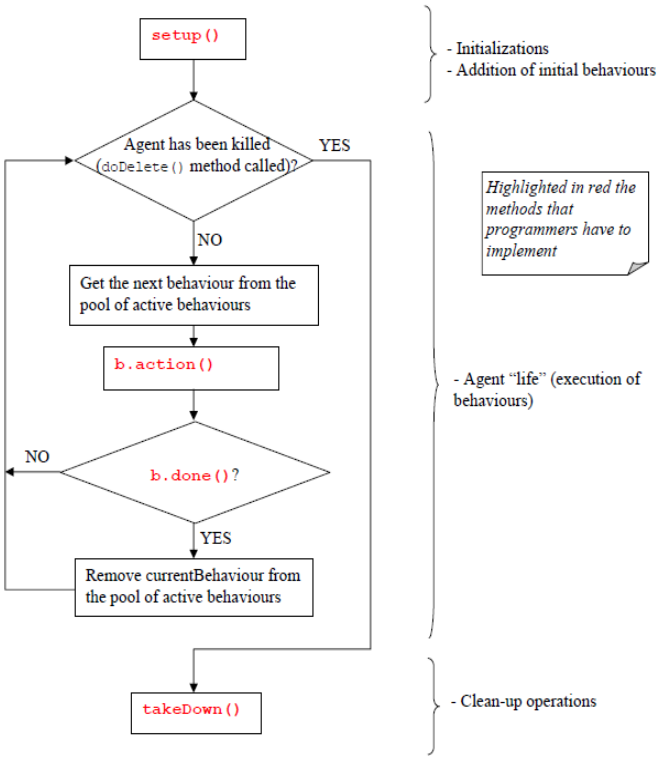

Figure 6, Agent behavior diagram. Source [24]

An agent behavior represents a task that an agent can carry out and is implemented as an object of a class that extends from jade.core.behaviours.Behaviour() [24]. There exist three types of behaviors; one-shot, cyclic and generic behavior. As seen in the flow chart in Figure 6, the skeleton of a typical agent program is composed of a setup() function where any

behaviors are added and any input argument to the agent is processed. The core of the agent's program is embedded within the behavior section. Each behavior class has an action() method which defines the operations to be performed. The action() method must be followed by a done() method which specifies whether or not a behavior has completed and removes the behavior from the pool of behaviors an agent is carrying out. When there are no behaviors available for execution, the agent thread goes to sleep in order to reduce CPU time consumption. The thread is awakened as soon as there is a behavior available for execution [24].

It is worth noting that many more agents could be created using the desired agent template, which is the advantage of using the JADE framework. For example, if a new generator (DER) is added to the PSH microgrid, its MAS software could inherit all the attributes (characteristics) and methods (operation and process) of the class DER agent. Therefore, it is not required to develop the control and/or the microgrid networking software for the new DER from scratch. This object-oriented approach not only reduces device software development time but also provides great plug-and-play ability that could allow a small network to be extended into a complex large-scale network [3]. The pseudo-code for each of the three agents will be delineated in chapter 3.

MACSimJX, the co-simulation engine

While Simulink is a very powerful simulation platform, it does not provide the tools to set up an agent framework. Fortunately, Simulink provides a work-around for adding functionality in the form of S-functions. This allows external programs written in other languages (in this case MAS written in Java) to be encapsulated inside the Simulink environment and run in their native language [22]. The issue arises as the S-functions are unable to handle multiple threads of execution - which is an essential attribute of MAS.

A software program called MACSimJX (Multi-Agent Control for Simulink program), is used as a gateway to allow data transfer between Simulink and the external MAS programs - with parallel processing capacity. It also provides the much needed synchronicity between the Simulink simulation and the MAS environment. The MACSimJX has a client-server architecture where the client is embedded in the Simulink's S-function and the server code is incorporated in a separate program [22]. The use of MACSimJX is not restricted for power system applications, for instance one agent application made use of the software to control the behaviour of a Boeing 747 in simulation [25].

As shown in Figure 7, the Simulink block contains the embedded MACSim S-function, which represents the interface between Matlab and the agent world (JADE). The S-Function takes in eight input signals: the current at the feeder, the power generated by the four synchronous generators and the power consumed

by the four loads. The initial goal will be to establish the bi-directional communication between the Simulink and the agent world. In other words, real-time data from the microgrid will be sent to the respective agent for analysis and control decisions would come back into Simulink in order to reconfigure the microgrid in the most power efficient way.

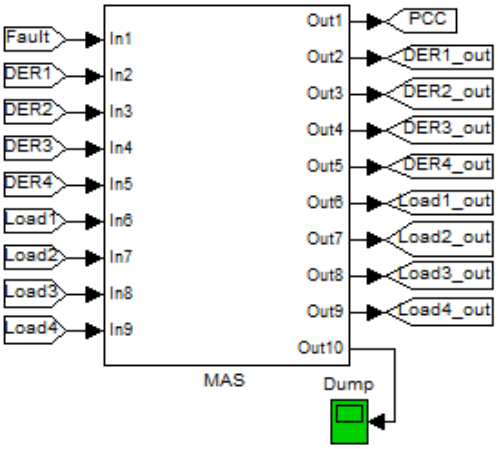

Figure 7, MAS block containing S-Function with MACSimJX

Typically, while the multi-agent system program is actively running, in a separate instance, the MACSimJX application is launched through the command prompt. Shown in Figure 8, the MACSimJX provides a simple graphical interface that enables the user to define the input and outputs of the MACSimJX S-function block. In other words, the definition will map the specified Simulink output signal to its specified agent implementation. The user also has the option to select the sampling frequency of agent system. In the current scenario, 100 kHz is used which is fast enough to handle faults (i.e. assuming a digital relay system operating at around 4 kHz).

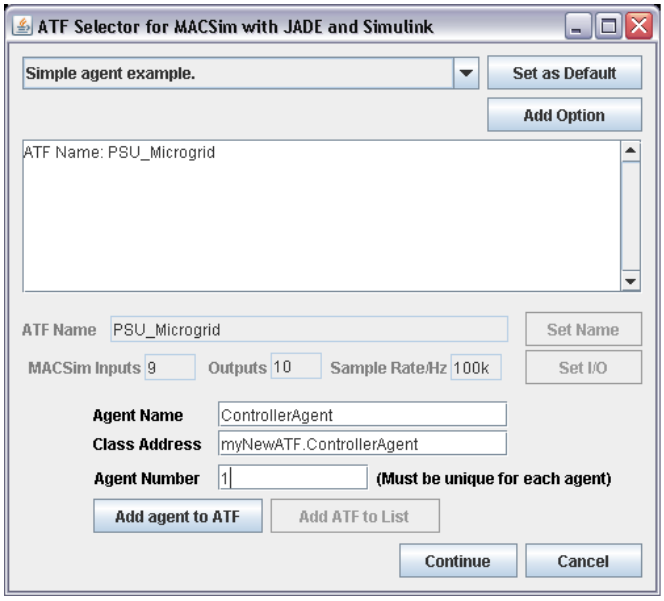

Figure 8, MACSimJX interface for gateway I/O definition

Figure 9 is a script file describing the agent definition process using MACSimJX. Basically, each of the 9 agents is created based on a .class file in the indicated class path. It could be observed that all DER agents are created using the same .class implementation, and the same applies for all load agents. This extensible software development approach reduces code development time for a large microgrid network composed of many DERs and loads.

```
ATF Name: PSU_Microgrid

Agent: 1
Name: ControllerAgent
Class path in package: myNewATF.ControllerAgent

Agent: 2
Name: DERAgent1_
Class path in package: myNewATF.DERAgent

Agent: 3
Name: DERAgent2_
Class path in package: myNewATF.DERAgent

Agent: 4
Name: DERAgent3_
Class path in package: myNewATF.DERAgent

Agent: 5
Name: DERAgent4_
Class path in package: myNewATF.DERAgent
```

```
Agent: 6
Name: LoadAgent1_
Class path in package: myNewATF.LoadAgent

Agent: 7
Name: LoadAgent2_
Class path in package: myNewATF.LoadAgent

Agent: 8
Name: LoadAgent3_
Class path in package: myNewATF.LoadAgent

Agent: 9
Name: LoadAgent4_
Class path in package: myNewATF.LoadAgent

MACSim inputs: 9, outputs: 10, sample rate (Hz): 100000
```

Figure 9, Agent creation using MACSimJX

Chapter 3. IMPLEMENTATION

This chapter will describe in detail the microgrid implementation using Matlab and touch upon the concept of islanding. Then the MAS implementation using JADE will be described and the pseudo-code for the Controller agent, DER agent and Load agent will be explained. The agent interaction at the moment of islanding will also be described. At last, the islanding simulation employing the MAS implementation will be presented.

Experimental setup

Preliminary MACSimJX software installation was not successful and it was determined that it was not optimized to execute on a 64-bit operating system. Therefore most of the JADE multi-agent system development was performed using the Toshiba computer while the Samsung computer was used for the Matlab Simulink microgrid simulation development. The final co-simulation was performed using the Toshiba computer which runs the MACSimJX that interfaced JADE with Matlab. Below is the description of the test setup:

Hardware

- Toshiba M100, Intel 1.66 GHz T2300, 980 MHz, 0.99 GB of RAM. Microsoft Windows XP, Home Edition, Service Pack 3, 32-bit OS
- Samsung RC512, Intel i5-2410M 2.30 GHz, 4 GB of RAM Microsoft Windows 7, Home Premium, 64-bit OS

Software

- JADE 3.6 (last update 06/18/2007)
- MATLAB R2009a, 32-bit
- MATLAB R2011a, 64-bit
- MACSimJX – MACSim with JADE extension, Pack A, version 1.7 (01/18/2011)

Microgrid monitor and control strategies

The combined co-simulation network is shown in Figure 10. The MAS function block takes in Simulink data, processes it using the MAS Java applications and outputs the control signal back into Simulink in order to activate the breakers.

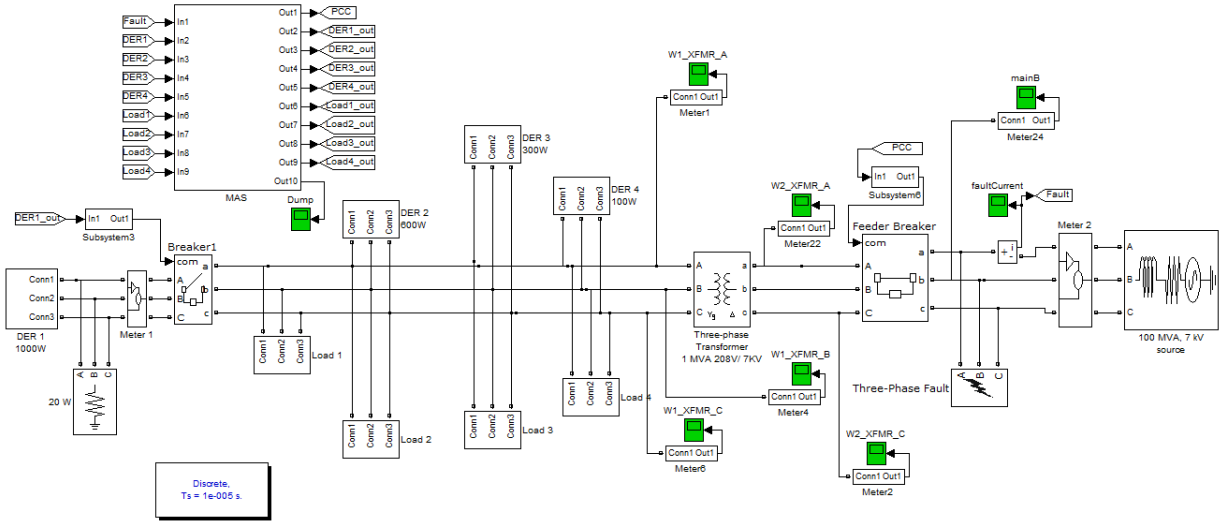

Figure 10, Co-simulation diagram

For the co-simulation, an islanding situation followed by a load balancing process and DER dispatch was implemented using the MAS. A single phase ground fault was introduced at the feeder on phase A. As seen in the Figure 11, there is a very high current surge - close to 1 million amps peak. If such high current level is not cleared, potential damage to the equipment connected in the microgrid could occur. Therefore the microgrid will be required to isolate itself from the utility, which is the motivation behind going from a grid-connected mode into the islanding mode. The other motivation is to ensure that the power flowing to all loads, especially mission critical loads (e.g. hospital or military installation) is uninterrupted.

Figures 11 and 12 respectively illustrate the simulated fault current surge at the utility feeder and the voltage waveform experienced on the 3-phase 208-volt network as a result of the fault. From Figure 12, it could be observed that the voltages on two of the three phases are operating at abnormal levels.

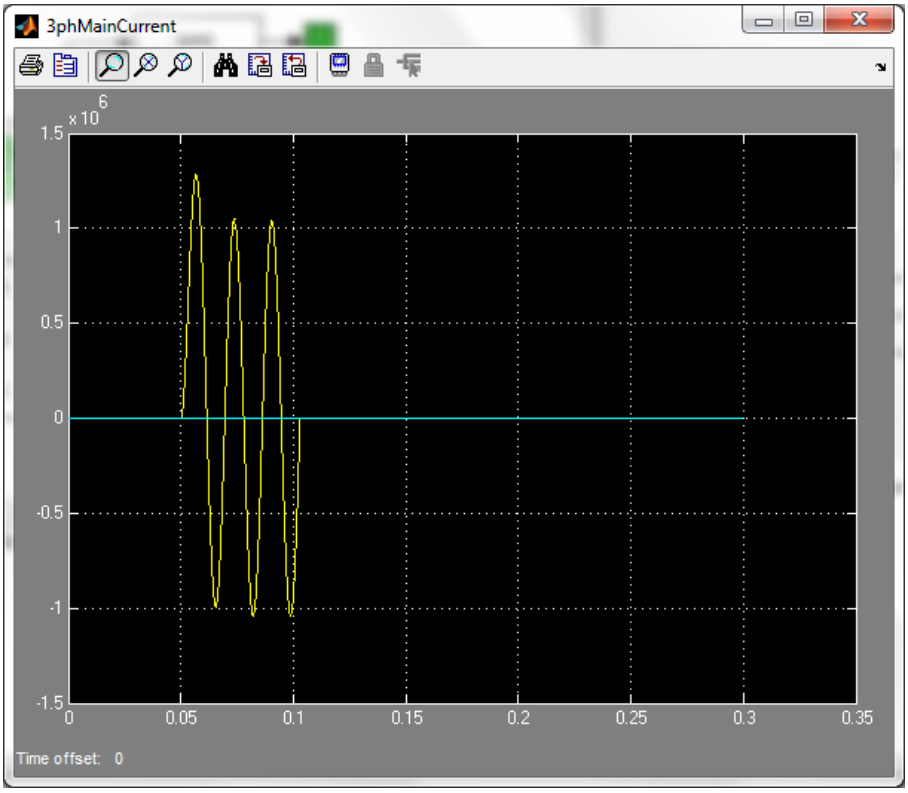

Figure 11, Ground fault current (A) between 0.05 and 0.1 second

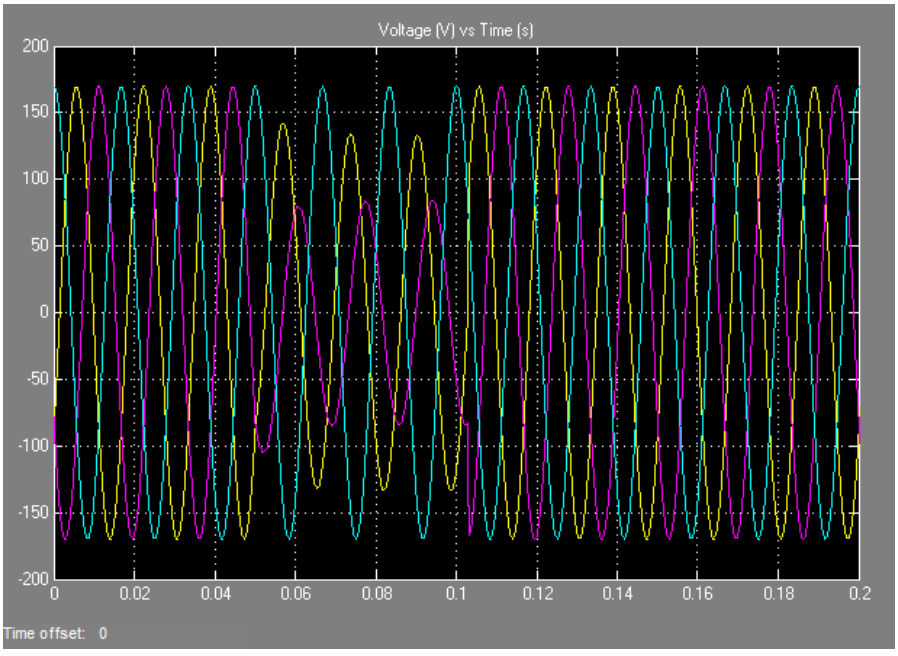

Figure 12, Line voltages (V) due to single-phase ground fault

In order to prevent the power transient from damaging the microgrid equipment, islanding will be required. To illustrate the concept of islanding, the feeder breaker is immediately opened following the fault by timing the control signal in Matlab. Shown in Figure 13, is the simulation response of a typical load bus if islanding is employed.

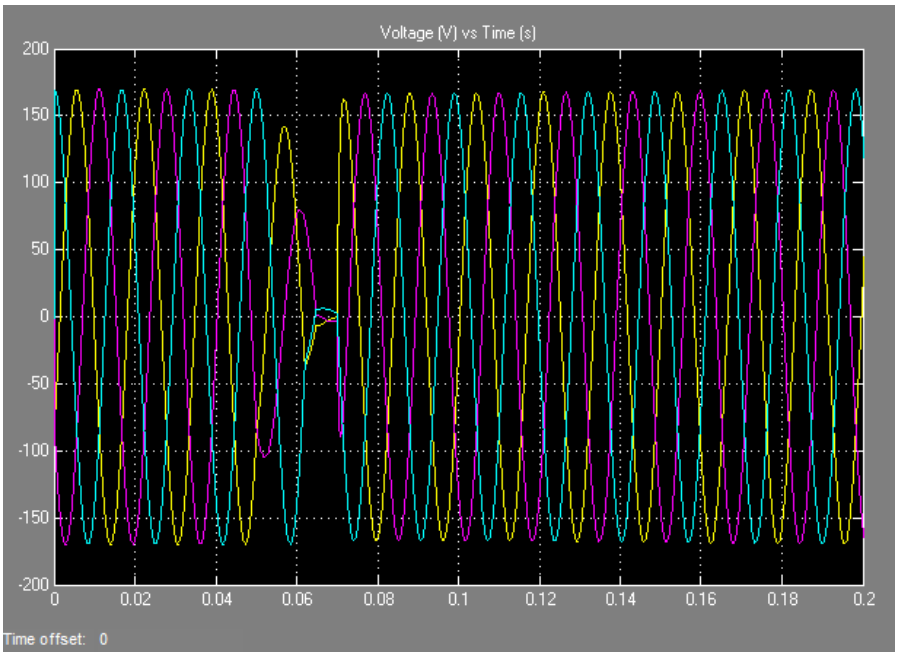

Figure 13, Line voltages (V) when islanding occurs

It could be observed that the voltages in the microgrid's 208V lines (i.e. 3-ph 120 Vrms) recovered much quicker than if they were left connected to the faulty mains (i.e. Figure 12). It should be noted that in this islanding case, it is assumed that the DER could produce sufficient power for the loads in the islanded configuration. However in a complex microgrid network where many DERs and loads are connected together and where power is often deficient, a network reconfiguration is required. Such reconfiguration during microgrid islanding could be a simple load balancing process where non-critical loads are shed so that the critical microgrid power demand could be met. A network reconfiguration might also entail the dispatching of DERs. Often times, DERs produce power at different costs depending on many factors such as fuel cost, solar irradiation, market incentives and other economic factors. Consequently, each DER has a unique cost function and it is often advantageous to dispatch the DERs to produce a specific amount of power in order to minimize cost.

For the purpose of this study, the MAS implementation will be designed to shed the loads from a predefined priority list and determining the ideal DER to be

dispatched. The co-simulation will showcase the conversation between the agents that will lead to a seamless islanding. More specifically, the feeder will be monitored for fault. If a fault is detected, the MAS Controller agent will issue a request to all the DER and loads registered to the microgrid service to provide their current status such as power produced, consumed, cost of producing power and other relevant data. With the given information, the controller will determine the ideal network configuration by sending out control signals to energize/de-energize the associated DER or load breaker(s).

Multi-Agent System programming

Natural representation of the world has previously been given as an advantage of object-oriented (OO) systems design, where entities in a system are modeled as *objects*. This has recently found favor with the power engineering community in standards such as the Common Information Model (CIM) and IEC 61850 [23]. The main benefit of the object approach is data-encapsulation. Meaning that the data structures which hold attributes of the object are hidden from external objects, yet they are indirectly accessible through method calls and standard interfaces.

The MAS Agent-based design provides another level of abstraction to the object approach by hiding the methods an agent can perform while the methods are still indirectly accessible through standard messaging interfaces [23].

The class AgentData encapsulates the attributes of a DER object and a Load object as shown in Figure 14:

```
public class AgentData {
	AID agentIDNum;
	int onlineStatus;
	int priorityNum;
	double minCap;
	double currentCap;
	double maxCap;
	double A;
	double B;
	double C;
}
```

Figure 14, Agent class

Each DER and Load object (agent) has an agent ID number and can let other agents know if whether or not its associated equipment is online and what is its

level of priority (i.e. critical load). In addition, it could communicate its minimum / current / maximum power level that it can or is producing or consuming (depending whether it is a DER or the Load). At last, the doubles A, B and C represent the coefficients of the cost function of the DER. These constants could be used in solving an optimal dispatch problem.

The pseudo-code for the Controller agent is delineated in Figure 15 as follows:

```
public class ControllerAgent extends UsefulAgentMethods

      Registers the "islanding" service with AMS

      ControllerBehaviour Cyclic Behaviour

            if data is received from Simulink
                  if current fault is detected
                        then send islanding signal to main breaker
                  else append data to a table

            if a message is received from DER agent
                  then save the AgentData data structure sent from DER
                  into a DER agent object array

            if a message is received from Load agent
                  then save the AgentData data structure sent from Load
                  into an Load agent object array


      DataProcessing Behaviour

            case 0:

            if fault is detected
                  then send a CFP message to all DER and Load agents

            case 1:

            if message received is from the recipient agents
                  then make sure they are PROPOSE messages

            case 2:

            if all DER and Load AgentData are received

                  then calculate total power produced
                  then calculate total power consumed

                  if total power produced > total power consumed
                        then output message "no network configuration is
                        required

                  else  then shed the non-critical loads by sending
```

```
                        an ACCEPT_PROPOSAL messages to non-critical

                        then send a REJECT_PROPOSAL loads that are not
                        chosen for shedding

                  if total power produced < total power consumed
                        then determine which DER is offline and dispatch
                        by sending an ACCEPT_PROPOSAL

                        then send a REJECT_PROPOSAL to DERs that
                        are not chosen for dispatch

            case 3:

            if message received is from the recipient agents
                  then make sure they are INFORM messages

      if case 3 is finished executing
            then DataProcessing Behaviour terminates
```

Figure 15, Controller agent pseudo-code

Communication and interaction between agents is facilitated through the use of service. As shown in the pseudo-code in Figure 15, the Controller agent has currently registered a service called “islanding” with the Directory Facilitator (DF). In other words, if an agent (in this case DER or Load agents) needs to use the “islanding” features of the Controller agent, the agent would request a search with the DF and obtain corresponding ID of the agent providing such service. Of course, the Controller agent is not restricted to the “islanding” service; more services could be added at any time in the future. An agent could have the option to make its service private and therefore to be able to discriminate who can use and access them.

The Controller agent is the trust center of the microgrid. It has a supervisory role and has control capability over other participating agents of the network. In the current islanding process, the controller has the responsibility of detecting a rapid current surge which is indicative of a ground fault. In the current scenario, if the detected current is greater than 0.5 million amperes, the Controller agent would send a control signal to the main breaker to transition the microgrid from a grid-connected mode to an island mode. Then, the Controller agent immediately executes the DataProcessing() behavior where a fault notification message is sent to the agents (DER and Load) that had subscribed to its “islanding” service.

Once the fault notification has been received by the DER and Load agents, the two agents will send their replies, containing their current power production or consumption respectively, back to the Controller agent. Additional DER and Load information such as priority, online/offline are also sent to the Controller agent.

The reason for such data gathering is to provide information regarding the DER and Loads so that the Controller agent could make a judicial decision about which load(s) could be shed and/or which DER could be dispatched in order to provide the required power to mission-critical loads in the microgrid.

The DataProcessing() behavior will keep listening for a *Propose* ACL Message coming from the DER and Load agents and makes sure that all proposals are received before moving forward. When all agent data had been received, the Controller agent will compute the total power produced and the total power consumed at the moment of the fault. The Controller agent will then use this "snapshot" of the microgrid power flow to determine whether or not to reconfigure the network in order to achieve sufficient power in the islanded-mode.

For example, if the total amount of power produced by the DER is larger than the total amount consumed by the loads; there is no need for load balancing. Otherwise, the Controller will request the different Load agents to disconnect their loads from the microgrid - in order of priority. In other words, the least important load is shed first. If after load shedding the total amount of power produced is still smaller than the power consumed, the Controller agent would then dispatch offline DER(s) to connect onto grid in order to compensate for the power deficiency. An *Accept_Proposal* ACL message is sent by the Controller agent to all the loads and DER considered for the load balancing and DER dispatch processes. For those that are not considered, a *Reject_Proposal* ACL message is sent.

The pseudo-code for the DER agent is shown in Figure 16:

```
public class DERAgent extends UsefulAgentMethods

      Registers the "DERPower" service with AMS

      DERBehaviour CyclicBehaviour

            if data is received from Simulink
                  then append the power data into a table

            if a message is received from another agent
                  then send the any relevant data to the sender


      ControlOrder CyclicBehaviour

            if the received message is a CFP message
                  then send a PROPOSE message to the sender by sending the
                  AgentData data structure


      ActionOrder CyclicBehaviour
```

```
            if the received message is an ACCEPT_PROPOSAL
                  then send a control signal to close the DER breaker
                  then send an INFORM message back to the sender
```

Figure 16, DER agent pseudo-code

In addition to listening for a fault notification message coming from the Controller agent, the DER agent also monitors its power production level. If a fault notification has been received, the DER will immediately send a *Propose* ACL message to the Controller agent. Furthermore, the DER's current power status as well as other pertinent information is also sent to the Controller agent. Once the information is sent, the DER agent will wait for an *Accept_Proposal* ACL message from the Controller agent in an event the DER had been chosen to be dispatched online. For the purpose of the study, the decision to dispatch a DER is based on its capacity and whether if it is available (i.e. offline). For future iterations of the MAS program, the cost function (i.e. coefficient A, B and C) that is sent to the Controller agent could be used to determine the actual amount of power to be produced if performing an optimal dispatch is desired. Once the DER had been dispatched, the DER agent will send an *Inform* ACL message to the Controller agent, essentially acknowledging that the DER had been successfully dispatched.

Very similar in structure to that of the DER agent, the pseudo-code for the Load agent is shown in Figure 17:

```
public class LoadAgent extends UsefulAgentMethods

      Registers the "LoadPower" service with AMS

      LoadBehaviour CyclicBehaviour

            if data is received from Simulink
                  then append the power data into a table

            if a message is received from another agent
                  then send the any relevant data to the sender


      ControlOrder CyclicBehaviour

            if the received message is a CFP message
                  then send a PROPOSE message to the sender by sending the
                  AgentData data structure


      ActionOrder CyclicBehaviour
```

```
if the received message is an ACCEPT_PROPOSAL
       then send a control signal to open the load breaker
       then send an INFORM message back to the sender
```

Figure 17, Load agent pseudo-code

Similar to the DER agent, the Load agent will constantly monitor and tabulate the power that its load is consuming while listening for any fault notification message from the Controller agent. Upon receiving the a fault notification, the Load agent will send its power consumption level, along with other pertinent information, to the Controller agent and then waits for an *Accept_Proposal* ACL message. If the message is received, the Load agent will first shed the load and then *Inform* the Controller agent that it had successfully done so. The agent messaging is encapsulated in the diagram in Figure 18. Each agent behavior is represented by a downward pointing arrow. The Controller agent has two behaviors (one cyclic and one generic), and the DER and Loads agents both have three cyclic behaviors.

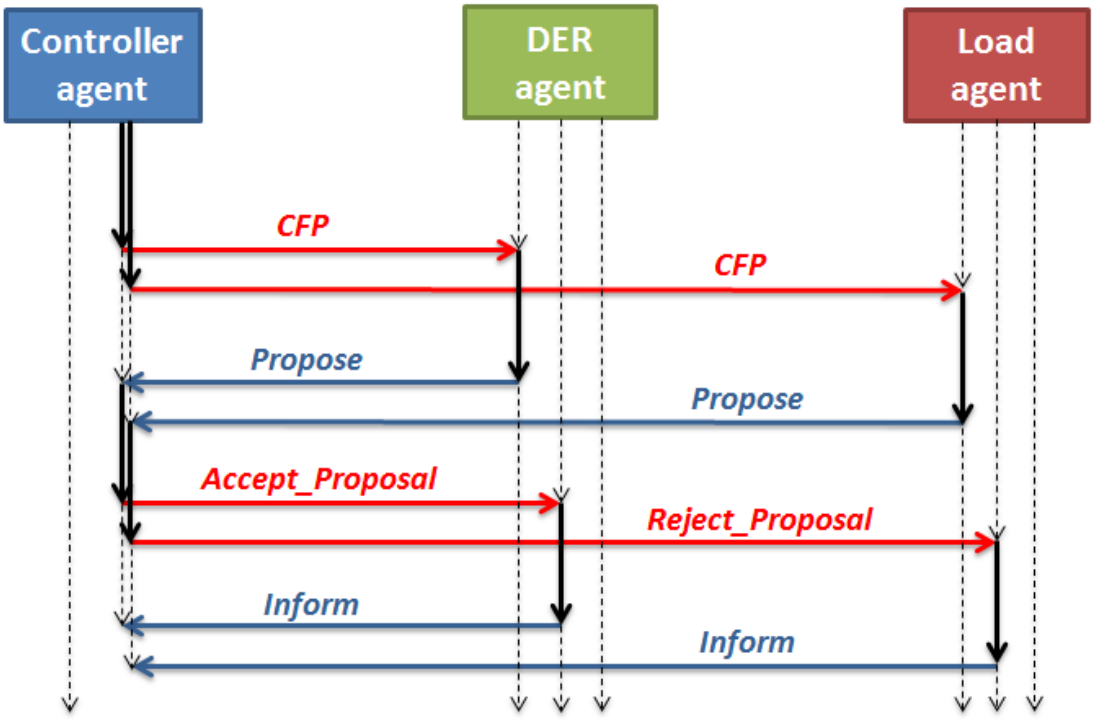

Figure 18, Agent conversation using ACL messages

Eclipse is used to develop the agent code. To test the MAS implementation, a simple dummy network was built, as seen in Figure 19. The network emulates the inputs for the DERs and loads (using step functions) and provides visualization at both inputs and outputs of the MAS. The dummy network played an important role during the MAS development process by cutting down on simulation time considering the large size of the actual microgrid network shown previously in Figure 10.

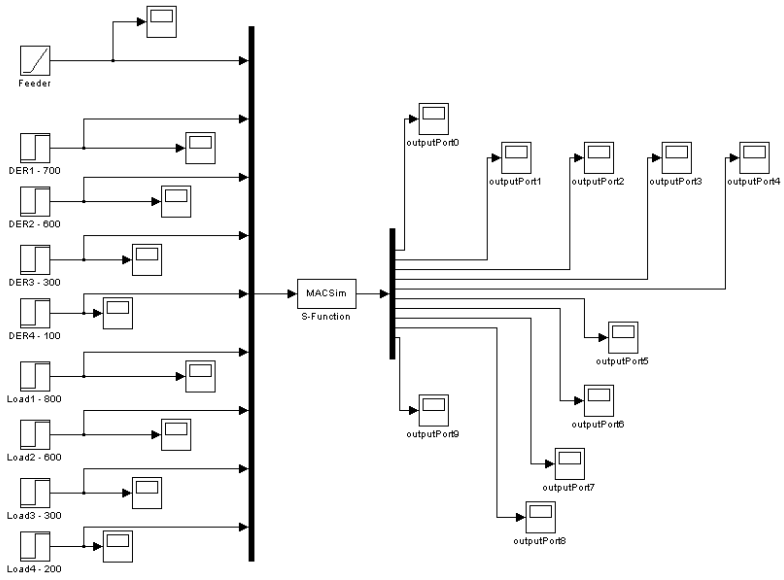

Figure 19, MAS test network for developmental purpose

Co-simulation result

Similar to the simulated fault scenario in Figure 13, a single-phase ground fault has been induced in leg A at the feeder. However this time, the islanding process and the load balancing will be completely achieved by the MAS. The simulation will last 0.1 second, just long enough to capture few cycles in order to illustrate the transient response during islanding.

The voltage profile seen at the three-phase line is shown in Figure 20. It could be observed that as the fault occurs around 0.06 seconds, there is a disturbance on two of the three phases. However this disturbance is short-lived as the microgrid quickly islands and performs the load balancing. In this particular case, two loads (load 3 and load 4) were shed off and generator 2 was brought online in order to

make up for the power deficiency (see case 3 of Chapter 4 for details of this islanding scenario).

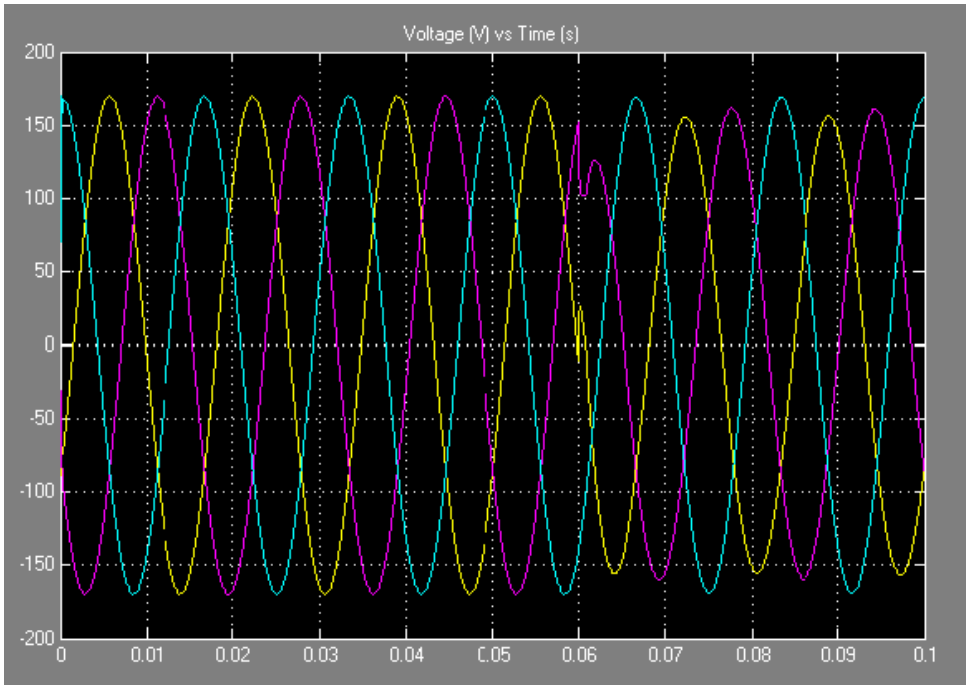

Figure 20, Three-phase voltage profile during islanding and load balancing

Obviously, it is important to keep the voltage transient experienced during islanding as short as possible. The performance of the islanding process is not quantified (i.e. overshoot, steady-state, etc). In this study, the goal is to illustrate the agent system process during microgrid islanding. However it could be concluded that the MAS was able to accomplish the islanding process, perform load balancing and DER dispatch while keeping the disturbance within a narrow window of about 0.003. The time during which the conversation between the agents took place could not be precisely quantified but it is indeed very short. More test cases illustrating the conversation between the agents will be delineated in chapter 4.

Figure 21 shows a sample of power data acquired by Load agent 2 and a sample plot generated in Microsoft Excel. It could be seen that the 600 W consumption was constantly being monitored by its associated agent.

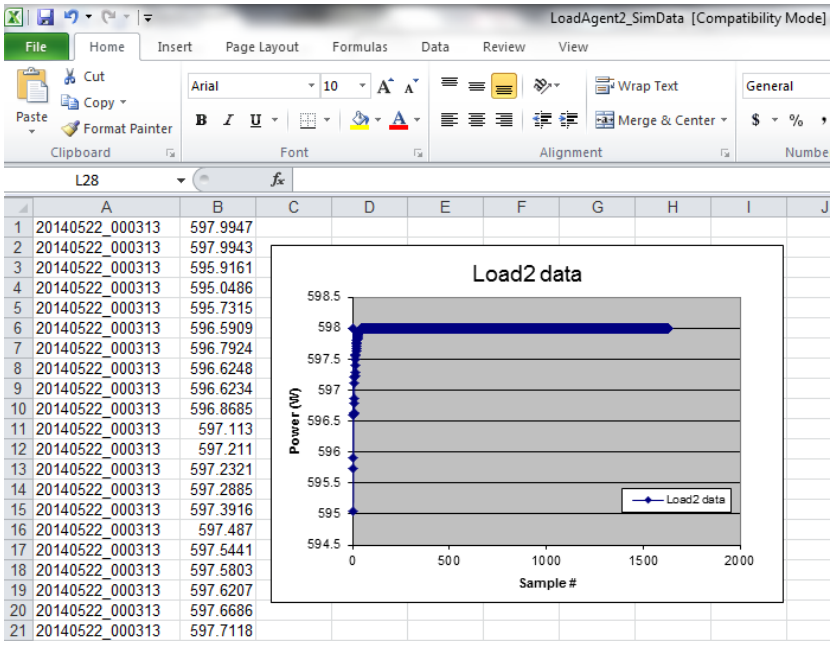

Figure 21, Sample plot of the load data acquired by a load agent

Chapter 4. ANALYSIS AND DISCUSSION

In this chapter, three islanding cases will be tested in order to illustrate the load shedding and DER dispatch processes. Improvement to the current MAS software will be discussed. Then the hardware-based microgrid work will be further re-iterated by introducing a ZigBee application. Finally the thesis project is concluded by recapping the discussed concepts.

Test cases

Different combinations of DER and loads are tested during the islanding process. Three cases will be used to illustrate in detail the load shedding and DER dispatching process. These important events are accompanied by output messages as seen in the command prompt window in Figure 22. All events are asynchronous because of the multi-threaded process; therefore, the order in which the output messages appear varies on each execution.

Case 1:

The first case illustrates a situation where the total power produced is larger than the total amount of power consumed. This is the ideal pre-islanding scenario where no action is required. Table 4 indicates the DERs and loads that are online and the respective priority number of each load (the lower the number, more important is the load). Figure 23 showcases the conversation seen by the Sniffer agent which is a built-in JADE agent responsible for displaying ACL messages between agents of the system. The Sniffer agent provides a network monitoring service by showing in real-time, the performative acts that are exchanged between the agents.

	DER1	DER2	DER3	DER4	Load1	**Load2**	Load3	Load4
Power (W)	700	600	300	100	800	**600**	300	200
Priority#	-	-	-	-	2	**1**	3	4
Status	-	Online	Online	-	-	**Online**		

Table 4, Load balancing case 1, critical load is in bold

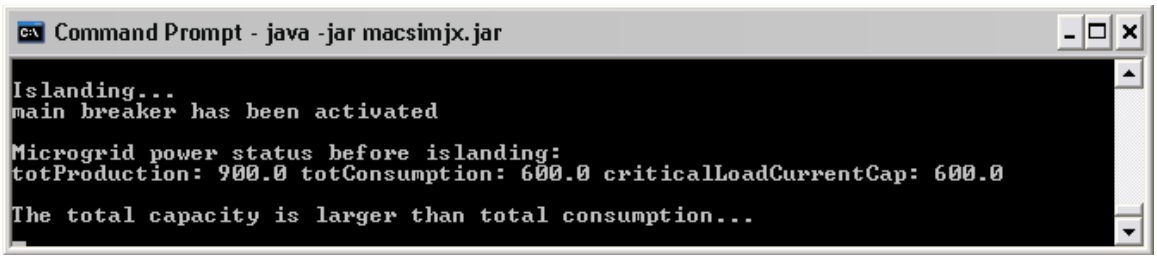

Figure 22, Result of islanding when capacity is sufficient

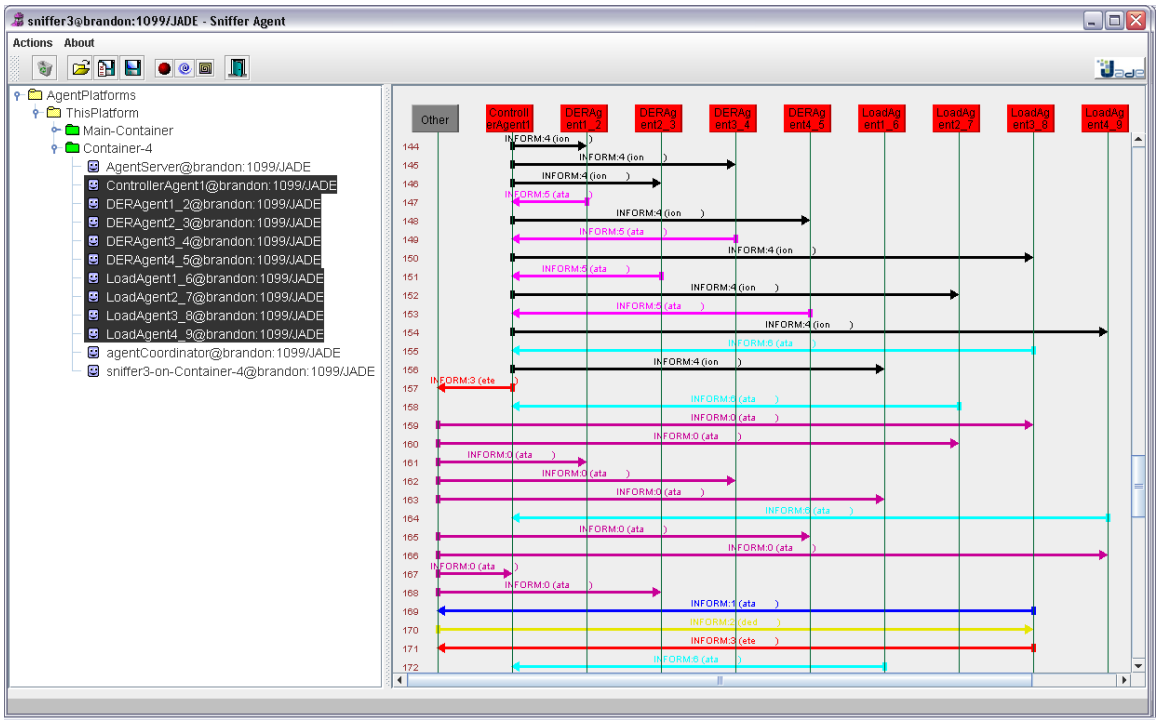

Figure 23, Conversation seen by the sniffer agent

Case 2:

The second case illustrates a scenario where DER 3 and DER 4 are online and are generating a combined power of 400 W. However, the online loads 2, 3 (critical load) and 4 are drawing a combined value of 1100 watts as shown in Table 5. Needless to say, upon islanding, the microgrid network would require reconfiguration in order to ensure sufficient power for the loads.

	DER1	DER2	DER3	DER4	Load1	Load2	**Load3**	Load4
Power (W)	700	600	300	100	800	600	**300**	200
Priority#	-	-	-	-	2	3	**1**	4
Status	-	-	Online	Online	-	Online	**Online**	Online

Table 5, Load balancing case 2

As shown in the command prompt in Figure 24, the Controller agent will command the load agents to shed their loads one by one, starting with the least critical one until the total power produced is larger or equal to the total power consumed. In this case, load 4 and load 2, with priority# 4 and #3 respectively, are shed. At the end, the load shedding process brought the total power consumption from 1100 watts down to 300 watts - allowing the microgrid to self-sustain in the islanded-mode.

Upon receiving the data structure sent from the load agents, the Controller agent sends an *Accept_Proposal* message to all potential Load agents whose load is considered for shedding. The Load agent accepts the shedding command and sends an *Inform* message acknowledging that the load will go offline. It should be noted that for simplicity's sake and since there is no real Contract Net interaction *per se*, the Load agent will always accept to shed off its load. Potentially in the future, an actual negotiation implementation could be employed. For instance, the load agents could propose an economic term and condition to which it will accept to go offline. The Controller agent on the other hand could have the ability to accept or reject (through the use of *Reject_Proposal*) any proposal made by the load agents.

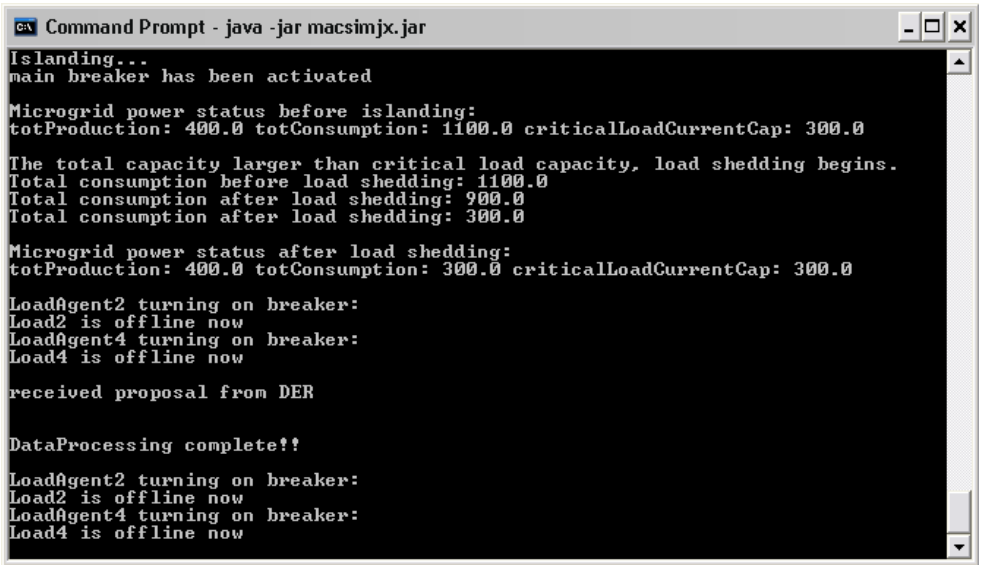

Figure 24, Result of islanding when microgrid has power deficiency

Similar to case 1, the conversation between the interacting agents are monitored and displayed by the Sniffer agent as seen in Figure 25.

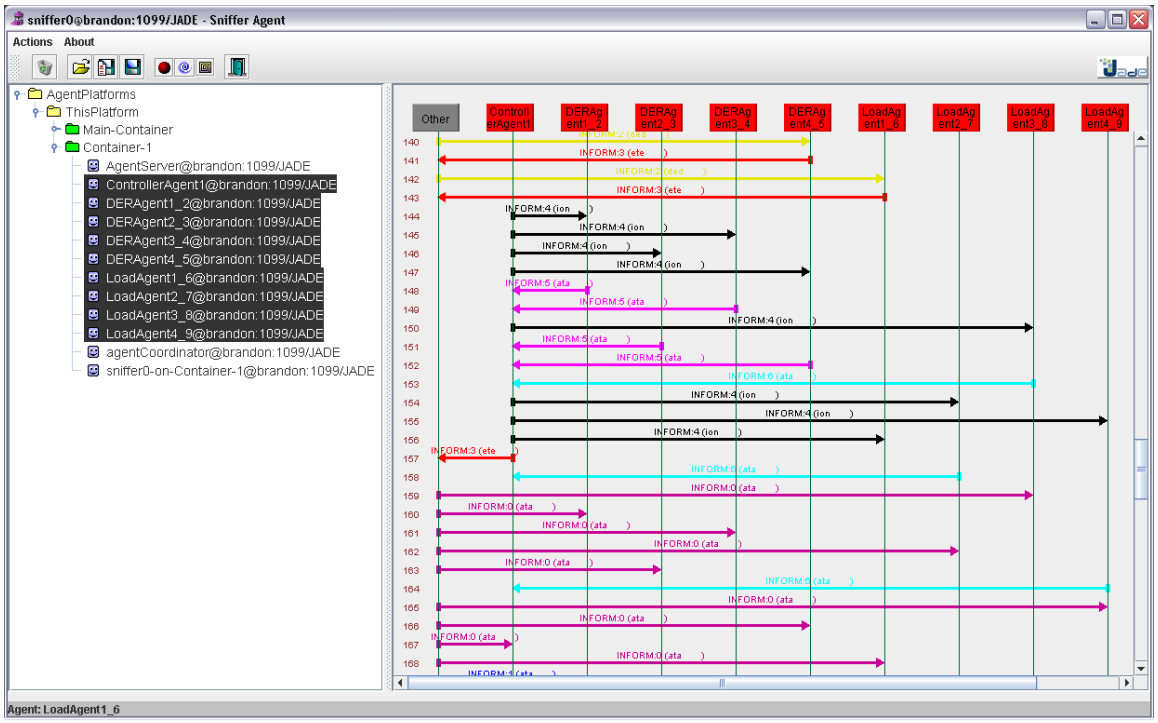

Figure 25, Conversation seen by the sniffer agent

Case 3:

The last scenario explores the process of DER dispatch. In a nutshell, if the microgrid is still power deficient (i.e. total power produced < total power consumed) despite performing the load shedding process, then available DER(s) could be brought online to compensate for the power demand. From the DER data structures, which are sent to the Controller agent by the DER agents upon detecting a fault upstream, the Controller agent enumerates all the different combinations of offline DER(s) that could be dispatched in order to compensate for the power deficiency. This process could be seen in Figure 26 under “possible combinations of DER”.

Also seen in Figure 26, the Load agents 3 and 4 are again called upon to shed their loads of 300 and 200 W respectively. However, the total power produced (400 W) is still 200 W short of the 600 W consumed. Consequently, the offline DER 2 is commanded to come online in order to ensure that the load 2 in the microgrid is properly supplied.

	DER1	DER2	DER3	DER4	Load1	**Load2**	Load3	Load4
Power (W)	700	600	300	100	800	**600**	300	200
Priority#	-	-	-	-	2	**1**	3	4
Status	-	-	Online	Online	-	**Online**	Online	Online

Table 6, Load balancing case 3

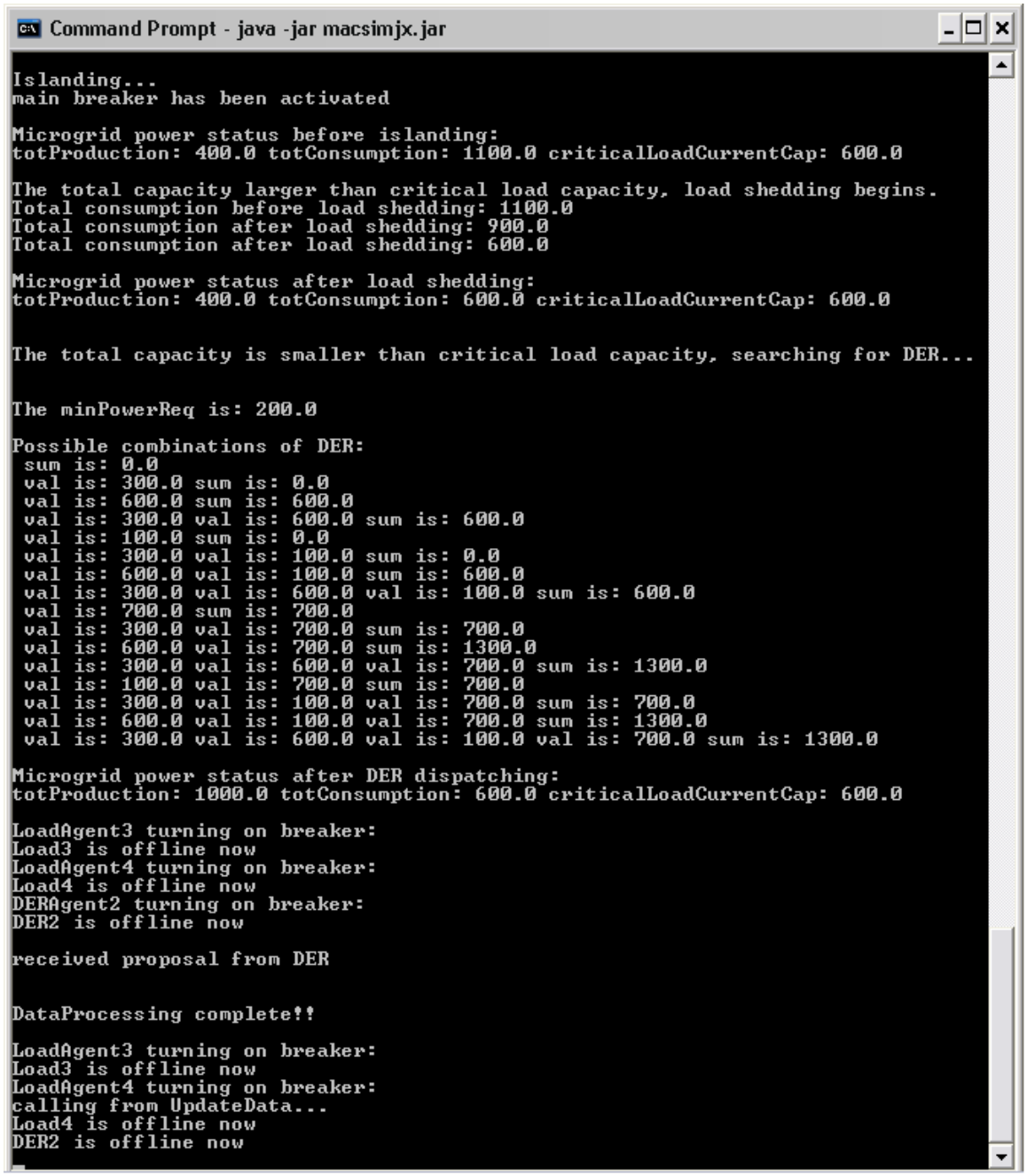

Figure 26, Result of islanding when microgrid has power deficiency

The conversation between the interacting agents is monitored and displayed by the Sniffer agent as seen in Figure 27.

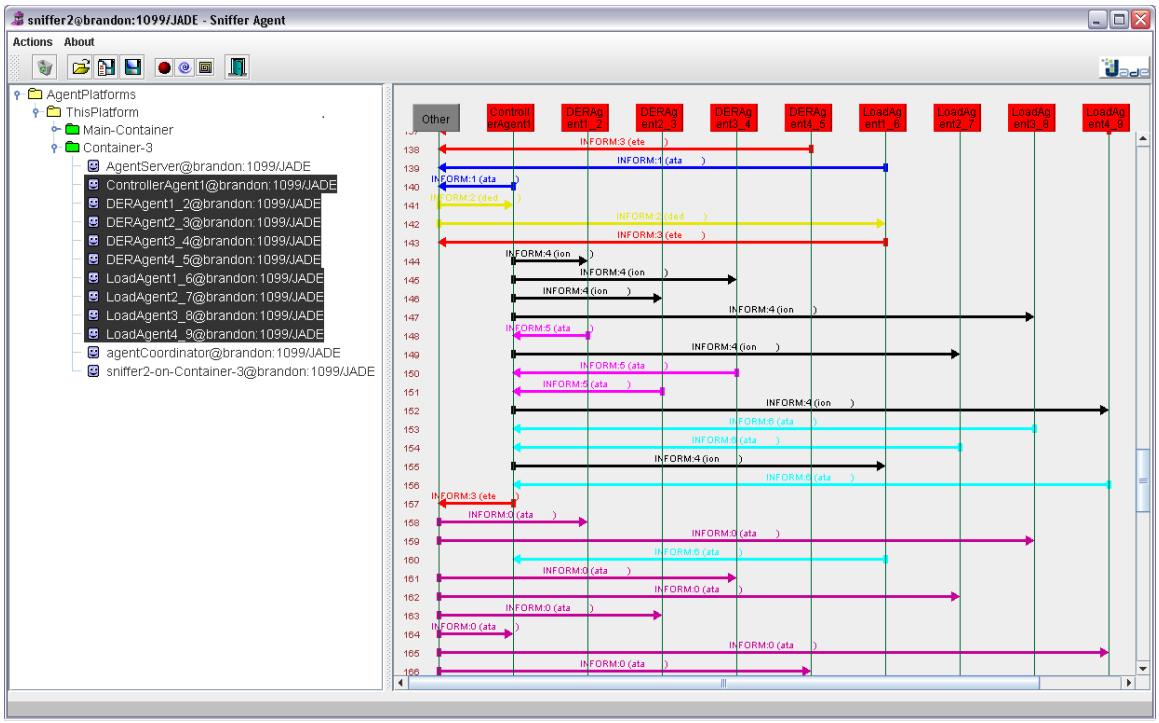

Figure 27, Conversation seen by the sniffer agent

The microgrid's simulation model is large and complex therefore the simulation time is lengthy; a 0.1 second simulation lasted about one hour. Furthermore, all nine agents are running on the same computer- therefore compounding to the computing bottleneck. For example, an extended simulation of two seconds took hours to complete. One solution was to disable local processes such as data appending or any unnecessary tasks. Consequently, it was observed that the simulation time could be shortened by as much as 30 percent. It should also be noted that the JADE's built-in sniffer agent encountered an error during its execution as seen in Figure 28.

Figure 28, Sniffer agent execution error

Base64 is an encoding/decoding technique, allowing for 8-bit data to be represented using only printable characters. It is believed that the error is probably the result of a missing conversion modules (The Base64 encoder / decoder was not originally a part of the Java language and there was a separate library that was required for this). According to the developer of MACSimJX, there is probably a conflict between the MACSimJX implementation and JADE's sniffer agent implementation. However it is believed that the conversation between the Controller and the DER/Load agents did occur simply because the Controller agent received the data structure it had requested from the DER/Load agents. Therefore the error is believed to be just an artifact of the conflict between MACSimJX and JADE implementations. The Sniffer agent captures in Figure 23, 25 and 27 are just meant to illustrate that the conversation took place between the agents as the content as well as the actual performative act might not be correct.

Improvement

The current MAS software is able to detect a fault and perform seamlessly the islanding, load shedding and DER dispatch processes. In addition to fault current detection, a frequency monitoring algorithm could also be implemented inside the Controller Agent. Furthermore, the restoration part of the simulation could be implemented - where the islanded microgrid is reconnected with the utility after the fault has cleared [6].

In this project, provision has been given for the Controller agent to achieve more advanced load balancing process such as using the polynomial of the cost function, sent from the DER, to perform an optimal dispatch. In the next iteration of the MAS co-simulation, it is envisioned that the Controller agent or another agent, whose task is to perform the optimal dispatch, would use the Lagrange multiplier to subsequently allow DER(s) to produce a specific amount of power. Another avenue of development is to exploit the Contract-Net protocol in an

actual bidding process. For example, in a power trading scenario where the DER(s) with the best price will produce power, a richer contract-net interaction could be implemented in order to handle the bidding, negotiation, awarding processes [7]. Furthermore, there could be room to investigate different algorithms used in solving assignment problems pertaining to the interactions between the buyer and seller agents such as auction algorithm [3, 26]. This will further enrich the interaction between DER and Load agents beyond just responding to request from the Controller agent.

As mentioned in Chapter 3, the three types of agents, currently making up the MAS are considered essential because of their important roles in the microgrid. It is envisioned that many other types of agents could be easily created and added to the MAS in the future. Furthermore, new agent interactions implemented through additional behaviors could be added to the existing agent code without effecting any changes to the overall agent system. Highly specialized agents could be created in order to accomplish specific tasks or solve specific problems. For instance, if a process requires the design of a generator stabilizer, then a "Power System Stabilizer agent" could be implemented in order to control and improve any transient stability of the synchronous generator. Similarly, a "Cost agent" could be created in a situation where an economic problem requires a solution. The type of agent and its role in the microgrid is left to the developer's discretion and the potential for agent exploitation and expansion is limitless.

Future hardware-based work

As the MAS program in this co-simulation project improves over the next iteration, when more behaviors / agents are added and more simulation scenarios are experimented, the thought process of incorporation of MAS into hardware should also be initiated. More specifically, the roles and location of the agents within the PSH microgrid should be determined.

For the moment, it is envisioned that each of the DER and load within the PSH microgrid will be interfaced with a computer or an embedded system. The Java Virtual Machine (JVM) and/or the JADE will be installed and running on these distributed computer systems.

The first step would be to port the Controller / DER / Load agent implementation from the co-simulation onto these computers. The JADE framework has advanced communication and agent management capabilities. In order to fully benefit from these underlying implementations, the agents should be first tested on different computer systems or on different networks. This topology would provide the necessary environment to test for agent communication and perform testing of agents that are physically located on different parts within the microgrid. For certain mission-critical applications written in other languages, it is understandable that the JADE could take on a data processing role rather than

that of a hardware control. For example, real-time data (or data from a database) could be piped into a Java-written agent for data mining purpose in order to forecast power consumption. Advanced computational models such as Artificial Neural Network (ANN) could be used to perform analytical and control functions. Furthermore, an area where JADE has seen a lot of development is in mobile applications. Therefore JADE-based agents could be developed to run on a smart phone to test for any mobile purpose.

Furthermore, it is envisioned that the MAS software could also be made to interface with a wireless ZigBee network where the agent software will be running on the end device computer as shown in the Figure 29.

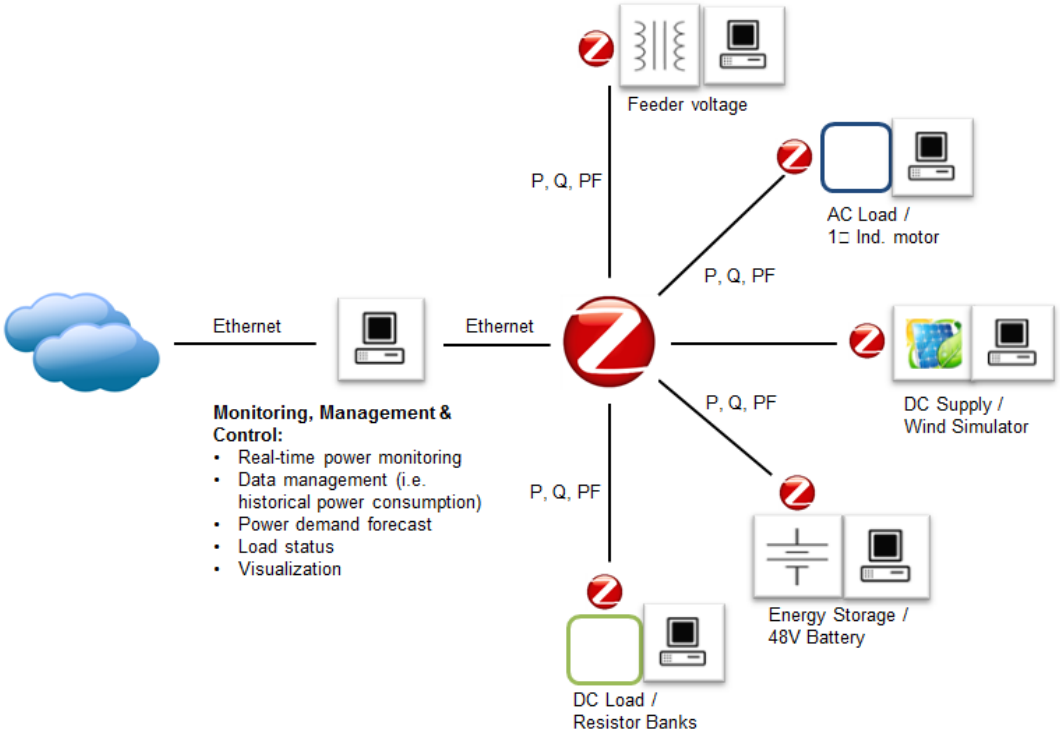

Figure 29, Proposed ZigBee network employing the MAS approach

The ZigBee standard, also known as IEEE 802.15.4 wireless standard, enables RF devices to operate on low-power, low data rate and on a secure communication network [27, 28]. The ZigBee wireless technology is currently used for smart energy system and home automation applications [27, 29, 30]. One particular ZigBee development platform is the Silicon Laboratories EM357 chipset. The EM357 ARM Cortex M3 processor combines a 2.4 GHz radio transceiver with a 32-bit microprocessor, Flash memory and RAM with powerful hardware supported network-level debugging features. This family of SoC

processors is highly integrated with the EmberZNet PRO software platform with mesh networking capability [30].

ZigBee shares many of the MAS attributes such as extensibility (i.e. plug-and-play) and autonomy (i.e. DLC) [28]. It is the communication solution of choice for a decentralized power network such the PSH microgrid. The EM357 was evaluated for the PSH microgrid development and a preliminary network was established along with basic communication functions between a coordinator node and two end-nodes. It is believed that the ZigBee network could add valuable monitoring capability to the PSH microgrid. More specifically, it is envisioned that the agents could be used as a ZigBee data aggregator [29] (i.e. data logging, event forecast, visualization, data conversion) or provide computing capability at the ZigBee node - as shown in Figure 29. The ZigBee specification defines the application and security layer specification of the technology, which is promoted by the ZigBee Alliance - a consortium of companies that draft ZigBee standards to ensure the interoperability of ZigBee-based consumer devices [19]. In terms of standardization, it is analogous to FIPA's agent system standardization where an open architecture could allow agents interaction regardless of the platform on which the agents have been created or executed.

Conclusion

The distributed nature of the smart grid could provide potential corrective actions in a case of a disturbance. However, smart equipment only offer one level of situational awareness, as smart decision-making is the more critical aspect of the resiliency [1]. The distributed decision-making must be in concordance with the overall objective of the collective grid network. In order to achieve the grid objective in the most cost efficient and effective way, the interactions between the participants of the grid should be resolved through the use intelligent communication, monitor and control technologies. While using the FIPA-ACL communication, it was shown in this project that the MAS approach could provide a better modeling approach therefore could enhance the interaction between microgrid entities.

The Multi-Agent System (MAS) is a good candidate for such application as evidenced by the islanding problem presented in this project. There exist many automated, intelligent and efficient systems or technologies which could be labeled as interpreted as "agents-like". Being able to distinguish an agent system from these advanced implementations is important so that potential advantages and benefit could arise in the designs. The ability of MAS to be flexible, extensible, and fault tolerant is often part of the justification for their use [23]. In this thesis, it was shown that by using the object-oriented approach based on the Java language, which is also platform-independent, the software development process could be greatly accelerated. This will become more apparent in a large network with many DERs and Loads.

Furthermore, it was shown that this extensible software approach further provided decentralization to the electric network by delegating tasks to different agents within an agent system. This method could eventually allow different DER and load equipment manufacturers to embed a generic microgrid agent program in their equipment which will provide the plug-and-play capability and promote equipment interoperability. On the contrary, in a centralized system the installation of any new component would require extra programming of the central controller [26].

Smart grid and microgrid encompass many hi-tech applications which are constantly evolving; it is therefore difficult to set in stone smart grid technology standards. This thesis also emphasized the importance of using an open and standardized MAS platform such as JADE. Jade could facilitate microgrid network expansion through the use of middleware implementation and by providing built-in agent development and management tools. One of the goals of this project is to raise awareness among power engineers who are currently (or in the future) deploying MAS or MAS-like technology that it is important to standardize the agent development work in order to be able to fully benefit the merits of a distributed network.

Appendix

Software development environment

Shown in Figure 30, is the typical MAS development environment illustrating its main components. In 1 is the Eclipse IDE where the agents are implemented. In 2 is the Matlab Simulink environment where the microgrid is designed and tested. In 3 is the command window used to compile the agent code. In 4 is the MACSimJX launcher window which also serves as the main output window for the agent process. In 5 is the JADE user interface.

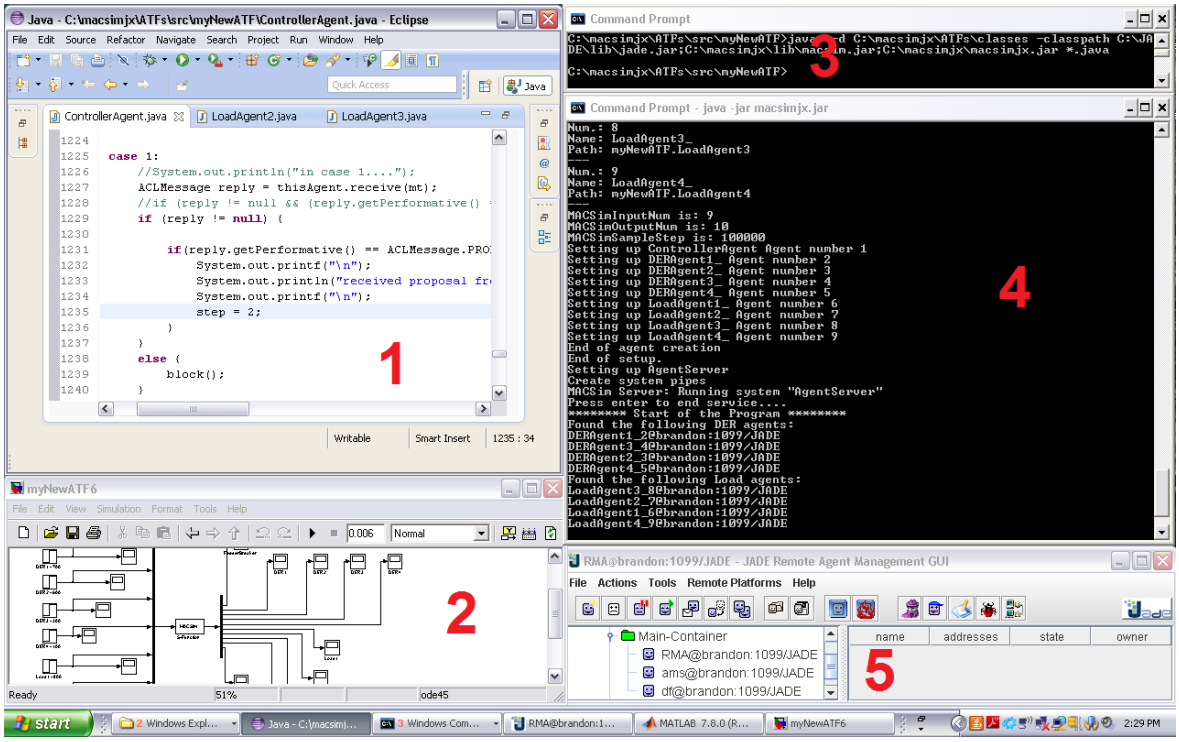

Figure 30, Co-simulation environment

Sample breaker control signal

During the islanding process, the signal that is used to control the opening and closing of the breaker is a pulse generated from the MAS program. For instance during the case 3 of islanding, the control pulses are generated by the MAS in order to open the main breaker (island) connecting the microgrid to the utility. Furthermore, two more control pulses are sent to open the breakers connecting load 3 and load 4 (load shedding). At last, a pulse used to close the DER 2 breaker (generator dispatch) is also generated by the MAS. These four control pulses are shown in Figure 31.

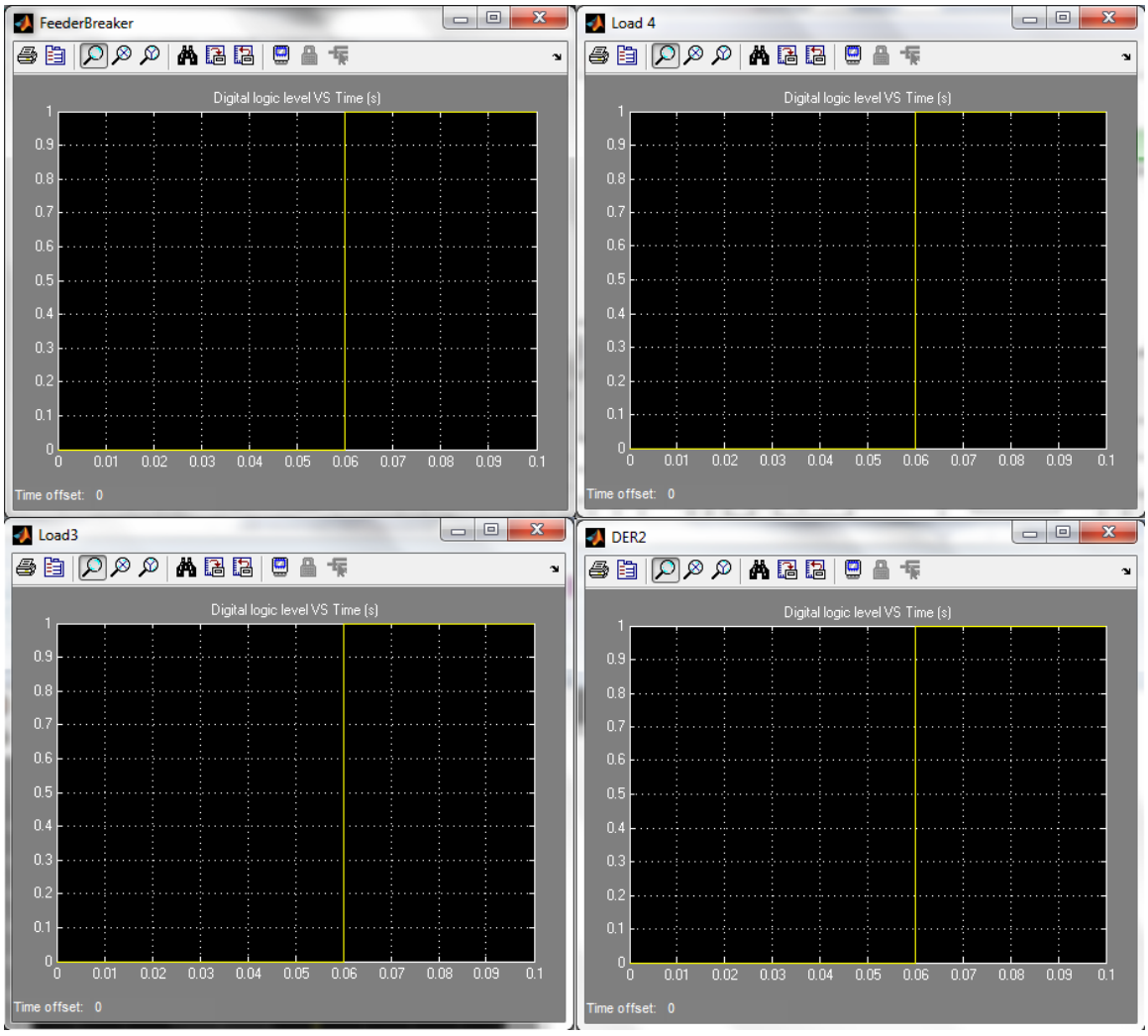

Figure 31, Pulse signals generated by MAS for breaker control

References

[1] Smart Grid System Report, US Department of Energy – 2009 [Online]. Available: http://energy.gov/sites/prod/files/2009%20Smart%20Grid%20System%20Report.pdf [Accessed: January 2014]

[2] García, Alvaro Paricio (CTO, IAPsolutions, Madrid, Spain); Oliver, Juan; Gosch, David. “An intelligent agent-based distributed architecture for Smart-Grid integrated network management”. Source: *Proceedings - Conference on Local Computer Networks, LCN*, p 1013-1018, 2010, *2010 IEEE 35th Conference on Local Computer Networks, LCN 2010*

[3] Kumar Nunna, H.S.V.S. (Department of Energy Science and Engineering, Indian Institute of Technology Bombay, Mumbai 400076, India); Doolla, Suryanarayana. “Multiagent-based distributed-energy-resource management for intelligent microgrids”. Source: IEEE Transactions on Industrial Electronics, v 60, n 4, p 1678-1687, 2013

[4] Smart Meter and Smart Meter systems: A metering industry perspective. An EEI-AEIC-UTC White Paper. © 2011 by the Edison Electric Institute (EEI). All rights reserved. Published 2011.

[5] Chatzivasiliadis, S.J., Hatziargyriou, N.D.; Dimeas, A.L. “Development of an agent based intelligent control system for microgrids”. Source: 2008 IEEE Power & Energy Society General Meeting, p 6 pp., July 2008

[6] Pipattanasomporn, M. (Virginia Tech-Adv. Res. Inst., Arlington, VA, United States); Feroze, H.; Rahman, S. “Securing critical loads in a PV-based microgrid with a multi-agent system”. Source: *Renewable Energy*, v 39, n 1, p 166-74, March 2012

[7] Foo. Eddy, Y.S.; Gooi, H.B.; Chen, S.X. “Multi-Agent System for Distributed Management of Microgrids”. Source: *IEEE Transactions on Power Systems*, May 21, 2014

[8] Karfopoulos, Evangelos L. (National Technical University of Athens, Athens 15773, Greece); Hatziargyriou, Nikos D. “A multi-agent system for controlled charging of a large population of electric vehicles”. Source: IEEE Transactions on Power Systems, v 28, n 2, p 1196-1204, 2013

[9] Consortium for Electric Reliability Technology Solutions (CERTS) [Online]. Available: http://certs.lbl.gov/ [Accessed: May 2014]

[10] B. P. Shaffer. UCI Microgrid White Paper [Online]. Available: http://www.apep.uci.edu/3/research/pdf/UCIMicrogridWhitePaper_FINAL_071713.pdf [Accessed: May 2014]

[11] Korea Smart Grid Institute (KSGI). Jeju Test-bed (2010) [Online]. Available: http://www.smartgrid.or.kr/eng.htm [Accessed: May 2014]

[12] Nagata, T. (Department of Electrical Engineering, Hiroshima Institute of Technology, Hiroshima, Japan); Sasaki, H. "A multi-agent approach to power system restoration". Source: IEEE Transactions on Power Systems, v 17, n 2, p 457-462, May 2002

[13] Kouluri, M.K. (Dept. of Electr. Eng., Banaras Hindu Univ., Varanasi, India); Pandey, R.K. "Intelligent agent based micro grid control". Source: *2011 2nd International Conference on Intelligent Agent & Multi-Agent Systems*, p 62-6, 2011

[14] Digra, R.K. (Electr. Eng. Dept., Banaras Hindu Univ, Varanasi, India); Pandey, R.K. "Multi-agent control coordination of Microgrid". Source: *2013 Students Conference on Engineering and Systems (SCES)*, p 5 pp., 2013

[15] Pipattanasomporn, M. (Adv. Res. Inst., Virginia Tech, Arlington, VA, USA); Feroze, H.; Rahman, S. "Multi-agent systems in a distributed smart grid: design and implementation". Source: *2009 IEEE/PES Power Systems Conference and Exposition (PSCE 2009)*, p 8 pp., 2009

[16] Rajendram, M.S. (Dept. of Electr. Eng., BHU, Varanasi, India); Pandey, R.K. "Multi agent control for two area power system network". Source: *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET 2012)*, p 134-7, 2012

[17] Abras, S. (LIG-Inst. IMAG, CNRS, Grenoble, France); Pesty, S.; Ploix, S.; Jacomino, M. "An anticipation mechanism for power management in a smart home using multi-agent systems". Source: *Proceedings of the International Conference on Information and Communication Technologies from Theory to Applications - ICTTA'08*, p 1466-71, 2008

[18] Abras, S.; Pesty, S.; Ploix, S.; Jacomino, M. “A multi-agent approach for the power management problem in smart homes”. Source: *Revue d'Intelligence Artificielle*, v 24, n 5, p 649-671, 2010.

[19] Hing Kai Chan (Norwich Bus. Sch., Univ. of East Anglia, Norwich, United Kingdom). “Agent-Based Factory Level Wireless Local Positioning System With ZigBee Technology”. Source: *IEEE Systems Journal*, v 4, n 2, p 179-85, June 2010

[20] McArthur, S.D.J. (Univ. of Strathlcyde, Glasgow, UK); Davidson, E.M.; Catterson, V.M.; Dimeas, A.L.; Hatziargyriou, N.D.; Ponci, F.; Funabashi, T. “Multi-agent systems for power engineering applications-part II: technologies, standards, and tools for building multi-agent systems”. Source: *IEEE Transactions on Power Systems*, v 22, n 4, p 1753-9, Nov. 2007

[21] F. Bellifemine, G. Caire and D. Greenwood. *Developing Multi-Agent Systems with JADE*. Chichester, England: John Wiley & Sons Ltd, 2007.

[22] Robinson B. C., Mendham P., Clarke T. “MACSimJX: A Tool for Enabling Agent Modelling with Simulink Using JADE”. Source: Journal of Physical Agents, Vol 4, No 3 (2010).

[23] McArthur, S.D.J. (Strathlcyde Univ., Glasgow, UK); Davidson, E.M.; Catterson, V.M.; Dimeas, A.L.; Hatziargyriou, N.D.; Ponci, F.; Funabashi, T. “Multi-agent systems for power engineering applications - part I: concepts, approaches, and technical challenges”. Source: *IEEE Transactions on Power Systems*, v 22, n 4, p 1743-52, Nov. 2007

[24] Giovanni Caire (TILAB, formerly CSELT). “JADE tutorial – JADE programming for beginners”. 30 June 2009. JADE 3.7

[25] Mendham, Peter (Intelligent Systems Group, University of York, Heslington, York, YO10 5DD, United Kingdom); Clarke, Tim. “MACSim: A simulink enabled environment for multi-agent system simulation”. Source: *IFAC Proceedings Volumes (IFAC-PapersOnline)*, v 16, p 325-329, 2005, *Proceedings of the 16th IFAC World Congress, IFAC 2005*

[26] Dimeas, Aris L. (National Technical University of Athens, Athens, GR 15773, Greece); Hatziargyriou, Nikos D. “Operation of a Multiagent System for Microgrid Control”. Source: IEEE Transactions on Power Systems, v 20, n 3, p 1447-1455,August 2005.

[27] Xiangyang Li (Inst. of Inf. Eng. &Tech., Ningbo Univ., Ningbo,

China); Weiqiang Zhang; Hu Jing. “Design of intelligent home appliance control system based on ARM and ZigBee”. Source: *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, p 260-3, 2012

[28] Jui-Yu Cheng (Dept. of Electr. & Electron. Eng., National Defense Univ., Tao-Yuan, Taiwan); Min-Hsiung Hung; Jen-Wei Chang. “A ZigBee-based power monitoring system with direct load control capabilities”. Source: *2007 IEEE/ACS International Conference on Computer Systems and Applications (IEEE Cat No. 07EX1688)*, p 6 pp., 2007

[29] Lingling Li (Sch. of Electr. & Inf. Eng., Xihua Univ., Chengdu, China); Weicheng Xie; Ziyang He; Xin Xu; Changmin Chen; Xiaorong Cui. “Design of smart home control system based on ZigBee and embedded Web technology”. Source: *Artificial Intelligence and Computational Intelligence. Proceedings of the 4th International Conference, AICI 2012*, p 67-74, 2012

[30] EMBER® EM35X Development kit User guide (UG110), Silicon Laboratories Inc [Online]. Available: http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf [Accessed: November 2013]