RCPC: A Multi-agent System for Coordinated Control of Power Electronic Converters in Microgrids

by

Maryam Nasri

MSS, University of British Columbia, 2009

B.Sc., University of Kerman, 1995

Dissertation in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Mechatronic Systems Engineering
Faculty of Applied Sciences

Approval

Name:	**Maryam Nasri**
Degree:	**Doctor of Philosophy**
Title:	***RCPC: A Multi-agent System for Coordinated Control of Power Electronic Converters in Microgrids***
Examining Committee:	**Chair: Woo Soo Kim** Assistant Professor

Mehrdad Moallem
Senior Supervisor
Professor

Herbert L. Ginn III
Supervisor
Associate Professor
Department of Electrical Engineering
University of South Carolina

Jason Wang
Supervisor
Assistant Professor

Amr Marzouk
Internal Examiner
Lecturer
School of Mechatronic Systems Engineering

Jahangir Khan
Senior Engineer
Stations Planning
BC Hydro

Date Defended/Approved: May 25, 2016

Abstract

The objective of this thesis is to describe the implementation of an innovative agent-based architecture of controllers for stand-alone DC microgrids. The controllers have to regulate voltage to the required level and manage energy flow in the system. In addition, they should maintain a deterministic time frame on the order of a few tens of milliseconds for a system with tens of power electronic converters with no limitation in the number of events which might happen concurrently. Optimal power sharing ensures minimum transmission and distribution loss while enforcing constraints such as generators' capacity limits. Multiple agents take part in the process to determine optimum power sharing for the converters. The thesis compares system complexity using numerical analysis of different distributed lookup algorithms based on defined metric values for a standalone DC microgrid including 32 converters. The numerical analysis results aid in choosing a publish-subscribe model as the most efficient and scalable solution for developing agent technology for standalone DC microgrids. Application of publish-subscribe agent-based control is presented for real-time coordination of power converters in a defined microgrid. To test the design, a sample DC shipboard microgrid with eight converters is used as a case study. Results of implementing the agent-based publish-subscribe control system using Java Agent DEvelopment Framework (JADE) are illustrated in the thesis. Simulation results affirm the accuracy of numerical analysis results. The results show that the upper time limit for task management is consistent and independent of the number of converters. The results of this research are published in seven articles, a list of which is included in Appendix D.

Keywords: Distributed algorithms, Distributed computing, Coordinated control, Energy storage, Load flow control, Microgrids, Multi-agent systems, Power systems, Publish-subscribe, Smart Grid, Supervisory control, Real-time power systems

To my first and forever teachers, my dear parents for their enduring love, advice, and sacrifices

To my loving husband, Reza, and son, Parsa, for their patience, encouragement, and support

Acknowledgements

Foremost, I would like to express my sincerest appreciation to my supervisor at Simon Fraser University of Canada, Dr. Mehrdad Moallem, and my advisor at the University of South Carolina, Dr. Herbert Ginn, for their exceptional supervision and leadership during the entire research. They generously shared their immense knowledge with no hesitation during different times of the research. Their positive reinforcement along with belief and patience led me to come up with this work. Without their invaluable guidance and advice this dissertation would not have been accomplished. I would like to thank you for your guidance and inspiration, which provided an invaluable experience that will help me for the rest of my life. I would also like to thank my examining committee members, Dr. Jason Wang, Dr. Amr Marzouk, Dr. Jahangir Khan, and Dr. Woo Soo Kim, for their time, interest, helpful comments, and insightful questions.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ACC	Agent Communication Channel
ACL	Agent Communication Language
AID(s)	Agent ID(s)
Ag(i)	Agent(i)
AMI	Advanced Metering Infrastructure
AMS	Agent Management System
ANSI	American National Standard Institute
API	(Application Program Interface) is a set of routines, protocols, and tools for building software applications.
ATF	Agent Task Force
BDI	Belief, Desire, Intention
CA(s)/CAg(s)	Converter Agent(s)
CB(s)	Capacitor Bank(s)
CHT	Constant Hash Table
CVR	Conservative Voltage Reduction
DF	Directory Facilitator Agent
DG(s)	Distributed Generator(s)
DHT	Distributed Hash Table
DMS	Distributed Management System
EDP	Economic Dispatch Problem
EMS	Energy Management System
FIPA	Foundation for Intelligent Physical Agents
Framework	In computer systems, a framework is often a layered structure indicating what kind of programs can or should be built and how they would interrelate. Some computer system frameworks also include actual programs, specify programming interfaces, or offer programming tools for using the frameworks.
GUI	Graphical User Interface is a type of interface that allows users to interact with electronic devices through graphical icons.
IA(s)	Intelligent Agent(s)
JADE	Java Agent DEvelopment Framework
KCP&L	Kansas City Power and Light Company

LTC(s)	Load Tab Changer(s)
PEBB	Power Electronic Building Block
MA(s)/MAg(s)	Middle Agent(s)
MACSim	Multi-Agent Control for Simulink
MACSimJX	Multi-Agent Control for Simulink Compatible with Java
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Problem
MIQP	Mixed Integer Quadratic Programming
MIQCP	Mixed Integer Quadratically Constrained Programming
MG(4/32)	MicroGrids (4/32 converters)
PA(s)/PAg(s)	Planning Agent(s)
Platform	A platform provides both the hardware and the software tools needed to run an application, be it a standalone program or one which has been built on top of a framework.
RCPC(4/32)	Real-time Coordinator of Power Converters (4/32 converters)
RLQs	Resource Lookup Queries
SA	Substation Automation
Smart Meter(s)	Smart Meter(s)
VVO	Volt-VAR Optimization
VVOE	VVO Engine
VR(s)	Voltage Regulator(s)
UML	Unified Modelling Language

Chapter 1. Introduction and Literature Review

The utility industry across the world is experiencing numerous challenges, including energy conservation, generation diversification, demand response, optimal deployment of expensive assets, and reduction of the industry's overall carbon footprint. These issues have led to the emergence of the smart grid concept in power distribution systems [1] since they cannot be satisfactorily addressed by the existing electricity grids. Conventional power systems convert only one-third of the fuel energy into electricity without recovering the waste heat. Regarding their hierarchical topology, the existing electricity grids suffer from domino effect failures. In addition, almost 20% of the generation capacity is used to meet the peak demand, while 8% of the output is lost in transmission lines [2].

The next generation of electricity grid, known as the "smart grid" or "intelligent grid," is expected to address the major shortcomings of the existing grid. In essence, a smart grid needs to provide utility companies with monitoring and pervasive control over their assets and services. A smart grid is required to perform energy transactions over power devices, and to be capable of self-healing. Smart grid is emerging as a convergence of power system engineering with information and communication technologies. To increase the capability of power grids, the first step is to incorporate enabling technologies in data management and communication at the distribution side. These ingredients place a layer of intelligence over the utilities' infrastructure by defining architectures, protocols, and standards towards the smart grid. The main capabilities of smart grid will be built on integration of applications located in the upper layer of the automation system. Some of these technologies are known as Distributed Management System (DMS) [3], Energy Management System (EMS), Substation Automation (SA), and Advanced Metering Infrastructure (AMI), which have been developed to make distribution networks more reliable and efficient [4].

Furthermore, the massive electrification in the developed world has exposed the distribution systems to a wide variety of unpredictable load profiles, with potentially negative and even unsafe levels of impact on the quality of service delivered to customers.

To counter such impacts, electric power utilities strive to employ new technologies to not only guarantee a certain level of quality of service (QoS) to their customers, but also to save energy, reduce distribution losses, and minimize operational costs. The smart grid replaces a hierarchical concept of generation, transmission, and distribution of power grids with an end-to-end intelligent and fully integrated environment. In these systems, distributed command and control strategies apply to all geographies, components, and functions. Smart Grid also can be defined as an interconnected network of microgrids with distributed control [5].

Microgrids are clusters of energy sources, storage systems, loads, local networks, real-time technologies, and load controllers that are organized to offer an energy solution for a community while connected to power grids or operated as an electrical island [6]. A microgrid may have numerous multi-functional power electronics converters connecting sources, loads, and energy storage systems to the bus. Systems in which converters are the interface between several energy sources and load centers have the possibility to direct the flow of energy if the control of converters is coordinated [7]. Figure 1.1 displays a sample islanded DC microgrid used in this research. Because microgrids are smaller and more flexible than power grids, different optimization algorithms were evaluated in both connected and islanded modes [8].

1.1. Research Objectives

In an autonomous microgrid, in order to make coordination among all converters, flow of power components need to be optimized. The optimization mechanism can include some functions such as minimizing power loss, balancing load flow, increasing stability and reliability, and reducing fuel cost. To achieve these functions, a comprehensive and coordinated control system is required to independently enable load control of each power component [9]. Since in the islanded mode of operation, voltage in a microgrid may suffer from low inertia and imbalance, the control system should include a method to control power quality at the load bus [10], [11]. In addition, for managing the coordination among local controllers, a higher control layer is required. The controllers have to regulate voltages to the required levels and balance load flow in all devices.

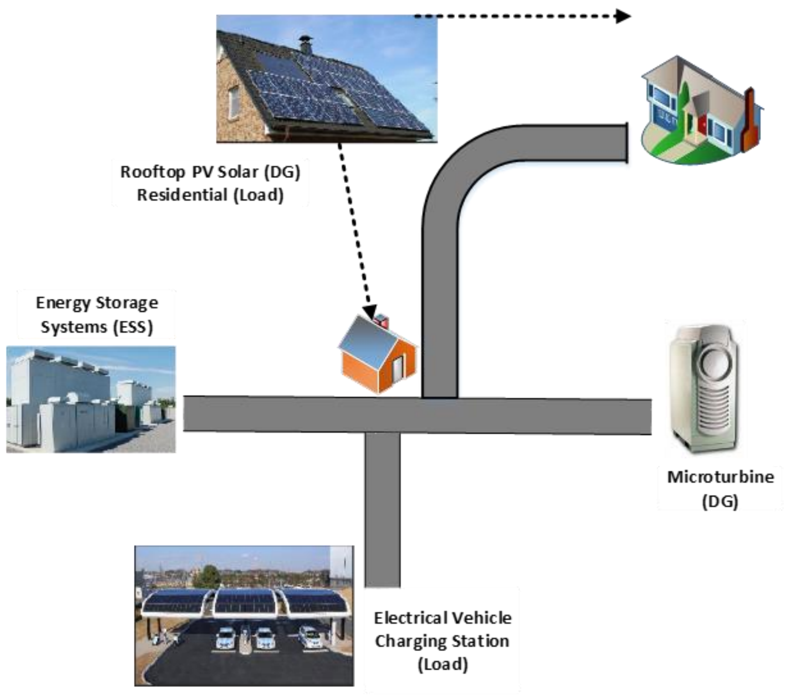


Figure 1.1. Example of a Microgrid Structure

Another important feature is the capability to maintain a deterministic time frame of a few tens of milliseconds for a system with tens of concurrently operating devices. A distributed coordination system approach can enable system-level control by avoiding single points of failure that are inherent in a centralized and hierarchical control system. The system should also be robust and expandable [12], [13].

Research performed in the area of distributed control indicates that a control based on agent technology has the potential to add extra advantages in terms of scalability, flexibility, and fault-tolerance compared with centralized structures [14]. It was shown in [15], [16] that the behaviour of the overall system using agent technology is considerably less affected by the failure of components. Another important advantage is that the modularity of the decentralized agent-based system can simplify the initial design stage and also facilitate the later addition of new elements to the system. With distributed processors, such systems can provide enhanced speed in terms of computation since there can be parallel processing of tasks with their own dedicated resources. This is particularly true when a power system is islanded and low inertia and where speed can be a critically important consideration for real-time systems.

In this document, a novel method for coordination of power converters in a medium-sized microgrid using agent technology is presented. The thesis contributions include application of the proposed agent technology in 1) Regulation of voltage to the required level in all converters; 2) Balancing load flow; 3) Obtaining a deterministic time frame for coordination on various number of converters in the sample microgrid. Since distributed systems deal with communication and computation of physically separate nodes, measuring message exchange is important and the number of message exchanges is defined as metric value. Applying different agent technologies for coordination among converters and comparing system complexity for different methods using metric values, help us to choose the most efficient and scalable agent-based solution for aforementioned controller actions.

1.2. Categorizing Microgrids Based on Their Main Specifications

Microgrids can be divided into different groups based on their specifications. There is no common definition of how to differentiate among different types of microgrids. The intersection of size (large or small) and grid connectivity (connected or remote) results in the following four main microgrid types (although the last type probably deserves further categorization):

1. Large grid-connected microgrids, such as those used in military bases and large campus applications, are connected to a traditional utility but are capable of operating in island mode. They have multiple generators and may have a substantial distribution system and sophisticated controls.

2. Small grid-connected microgrids have a single Distributed Generator (DG) which is supplemented with storage and renewable energy resources, as appropriate. Grid-connected microgrids are typically used in areas with unreliable grids where a backup generator is frequently used. One may not consider these to be microgrids, but they own common characteristics with other types of microgrids. Since they can independently operate in the grid, they are a valid and potentially important form of microgrid.

3. Large or medium remote microgrids, such as those used in island utilities, have multiple generators and a substantial distribution.

4. Small remote microgrids usually have one DG, or instead of DG they have sources such as battery-based storage or fuel cells17. Some of the very small ones may have DC distribution. Innovations in billing and payment methods could greatly enhance the potential to use these types of microgrids [18].

Shipboard microgrids are often categorized as large size, remote microgrids, and a smaller size of the main system is considered as a case study. An optimization of power flow will be performed that minimizes system losses.

1.3. Control Technologies in Microgrids

Different control architectures for power grids have widely been in use based on central and hierarchical methods. Considering their higher efficiency and reliability, recently decentralized and fully distributed controllers such as intelligent controllers have been beginning to appear [19]. Intelligent controllers are rarely used due to complications in their control coordination. In addition, less complicated controllers such as proportional-integral-derivative (PID) are generally preferred, although they have some disadvantages such as the difficulty in controlling nonlinear features. Table 1.1 compares some of the main specifications of centralized and distributed controllers in a medium-sized distributed microgrid [20].

While the central controller may be most suitable for small-scale systems, distributed control systems provide a more reliable strategy to control the local devices in a large-scale system. As mentioned in section 1.2, the size and status of power systems are the main parameters for choosing a suitable control method (Figure 1.2).

Table 1.1. Comparison between central and distributed control systems for a medium-sized distributed microgrid

	Central Control System	Distributed Controlled System
Pros	Control algorithm is relatively simple	Relieved the computational load for a single controller Ease of heavy data exchange demand Single point of failure will not necessarily affect the others
Cons	Computational limitation of central controller Communication limitation of central controller Single point of failure will affect the entire system	Only part of the system states are available to each distributed controller Normally need complex algorithms and designs
Usage	Normally more appropriate for small systems with simple control	More appropriate for large-scale systems which need sophisticated control

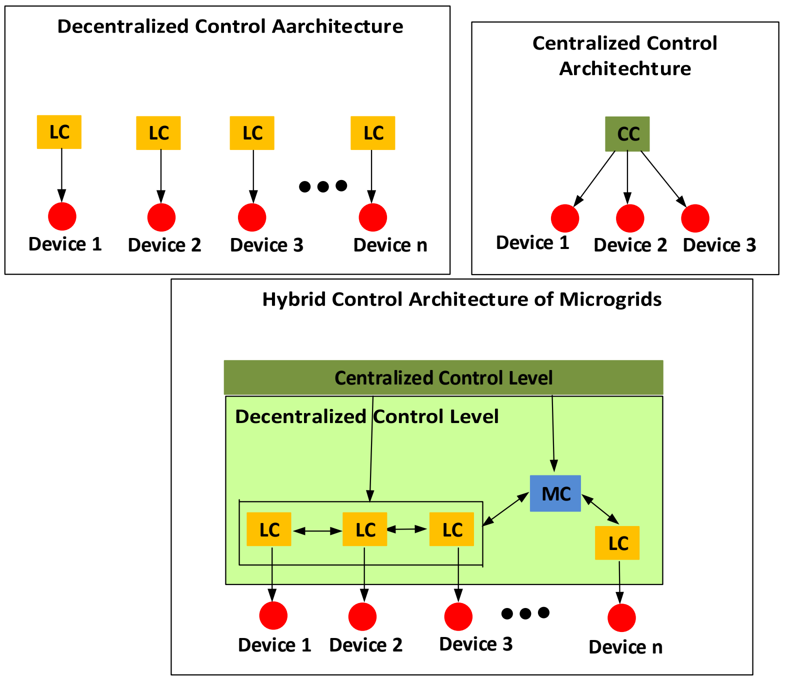

Decentralized Control Aarchitecture
LC
LC
LC
LC
Device 1
Device 2
Device 3
Device n
Centralized Control Architechture
CC
Device 1
Device 2
Device 3
Hybrid Control Architecture of Microgrids
Centralized Control Level
Decentralized Control Level
MC
LC
LC
LC
LC
Device 1
Device 2
Device 3
Device n

Figure 1.2. Demonstration of Different Control Structures (LC: Local Controller, MC: Middle Layer Controller, CC: Central Controller)

Smart grids utilize a combination of centralized and decentralized (distributed) control systems [21]. Centralized control systems have the highest performance in small-scale power networks and deliver power in one direction (i.e., from substation to loads). Nowadays, evolution of some routines in power distribution systems such as distributed power storages and generation requires deploying smart control systems. Distributed command and control systems can use agent technology to implement their control strategies. It is worth mentioning that most traditional power control systems act preventively or reactively to events, in which the current control systems add active control options to their control strategies [22].

The growth in shared information resources requires information systems that can be distributed on a network and interoperate with other systems. Such systems cannot be easily realized with traditional software technologies because of the limits of these technologies in coping with distribution and interoperability. The agent-based technologies may be an answer to facilitate the realization of such systems because they were invented to cope with distribution and interoperability [23].

1.4. Agent–based Technologies and Multi-agent System (MAS)

An agent is a software (or hardware) entity that is situated in an environment and is capable of autonomously reacting to changes in that environment. An agent is designed to deliver its design objectives based on dynamic environmental requirements. The environment can be physical (e.g., a protection switch), or computing space (e.g., a software program) [24], [25]. MAS is a system comprised of two or more agents or Intelligent Agents (IAs) [26]. Agents may or may not be able to communicate with each other but IAs must communicate together. MAS are mostly designed, implemented, and used in distributed systems. The combination of three technologies including web services, grid computing, and intelligent systems are used in MAS.

Distributed systems have many interesting technical aspects such as decentralized control, self-organization, adaptation, and scalability. There are currently many projects aimed at constructing distributed applications and understanding more issues and requirements [27]. One of the key problems in the control of large-scale distributed applications is the provision of efficient algorithms for object location and routing within the network. An efficient control algorithm seeks to minimize the distance messages travel, according to the scalar proximity metric, such as the expected number of routing hops and the number of messages exchanged among system nodes. Multi-layer control technologies are one of the most common solutions for distributed systems.

Considering multi-layer control technologies, three types of distributed control architectures are commonly defined for power systems: hierarchical (functional), heterarchical, and hybrid architectures. Hierarchical architecture requires communication among N control layers, which are connected to each other as a client-server. There is no direct communication between modules of the same level. While this architecture proposes a hierarchical modeling methodology for real-time scheduling, its feasibility and optimality are not proven. The heterarchical control architecture is based on full local autonomy (distributed control) resulting in a control environment in which autonomous components co-operate in order to reach global objectives through local decision making [28]. In a distributed heterarchical model where each agent represents an individual resource, each agent individually implements these low-level control algorithms for all the resources they represent. Duffie in [29] explains about some of the other advantages of heterarchical architectures including reduced complexity, increased flexibility, and reduced costs. Figure 1.3 displays a model of different control technologies applied to a microgrid.

The quality of distributed computational node technologies are compared based on some parameters such as scalability, fault management, and convergence. Table 1.2 compares five different technologies for distributed systems based on the aforementioned factors. As seen, agent control technology is a combination of advantages of centralized, distributed, and hybrid controls technologies.

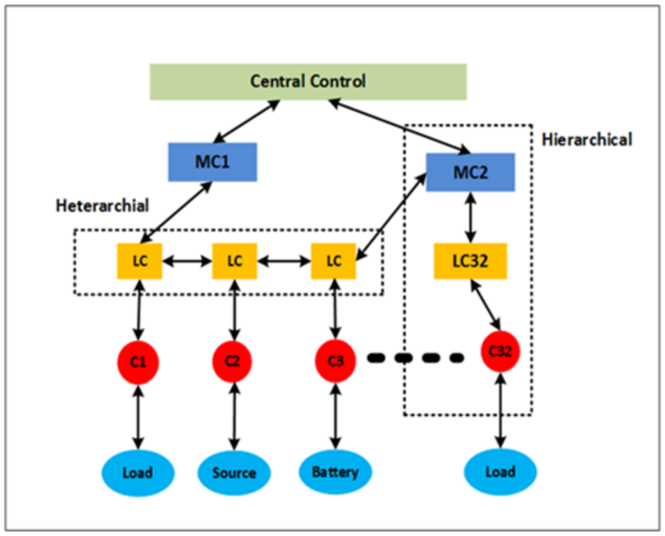


Figure 1.3. A Model of Different Control Technologies Applied Microgrids; Hierarchical, Heterarchical, and Hybrid (Hierarchical + Heterarchical) Control Technologies

Table 1.2. Comparison between control technologies

Reviewed Control Technologies	Flexibility of Adding and Removing Nodes (Scalability)	Ability of Fault Management (Self-Healing)	Real-time Convergence Rate
Centralized	Low	Low	Medium
Hierarchical	High	High	Medium
Heterarchical	Medium	Medium	Medium
Hybrid	High	High	High
Agent Tech.	High	High	High

A multi-agent control system provides a higher degree of efficiency and reliability which works within a real-time and fully distributed system, where each node can be connected to other nodes at the same or different layers. These communications can be scheduled *on-demand* based on their tasks such as self-healing and data mining. It could define valid ranges of data, detect out-of-range data, and transfer the data and fault messages to the destination [30]. Multi-agent systems are distributed networks, containing intelligent hardware and software agents, that work together to achieve a global goal. To enable agents to work together in order to achieve power system objectives. As shown in Figure 1.4, MAS allocates a local view of the power system to each agent, empowering a group of agents to control a wide distributed power system. Furthermore, MAS has an event-driven real-time architecture where a group of agents work together to optimize a parameter of the distributed network. The system needs to be flexible in adding and removing distribution nodes, and it is desirable to self-heal after fault occurrence.

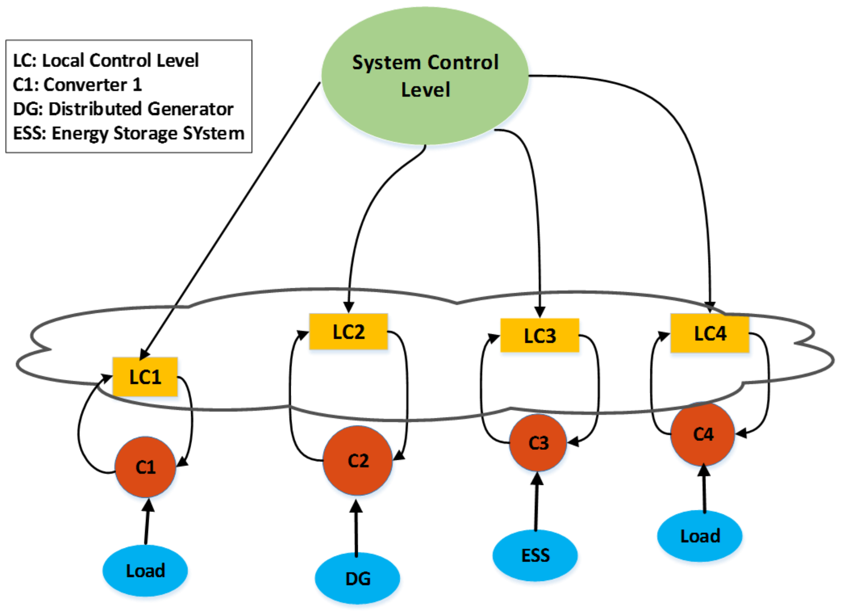


Figure 1.4. An Schema of Agent-based Control Technology

1.4.1. Mathematical Model of Agents

Agents and MAS are mathematically modeled based on their definitions. Assume the environment may be in any of a finite set E of discrete, instantaneous states:

$$E = \{e_0, e_1, e_2, \dots\} \tag{1.1}$$

Agents have a range of possible actions (Ac), which transform the state of the environment:

$$Ac = \{\alpha_0, \alpha_1, \alpha_2, \ldots\} \tag{1.2}$$

A run of an agent is a sequence of an interleaved environment states and actions:

$$r: e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \cdots e_{u-1} \xrightarrow{\alpha_{u-1}} e_u \tag{1.3}$$

R is the set of all possible finite sequences:

$$R = \{r_1, r_2, r_3, \ldots\} \tag{1.4}$$

An environment Env is dependent to triple parameters:

$$Env = \langle E, e_0, \tau \rangle \tag{1.5}$$

where:

- E is a set of environment states
- $e_0 \in E$ is the initial state of Env
- And τ is a state transformer function represents behavior of environment. It maps a run-ending action to a set of environmental states. Considering that environments are history dependent, and non-deterministic:
  - $\tau(r): R \to P(E)$
  - If $\tau(r) = \emptyset$, there are no possible successor states to r, it means that the system has ended its run.

Agent is a function which maps runs to actions. An agent makes a decision about what action to perform based on the history of the system that it has witnessed to date:

$$Ag: R \rightarrow Ac \tag{1.6}$$

MAS will be the set of all agents:

$$MAG = \{Ag_1, Ag_2, Ag_3, \dots\} \tag{1.7}$$

Agent control loop has the following steps and shown in Figure 1.5:

1. Agent starts in some initial internal state i_0
2. Observes its environment state e
3. Generates a percept $see(e)$
4. Internal state of the agent is then updated via *next* function, becoming
$$next(i_0, see(e)) \tag{1.8}$$
5. The action selected by the agent is
$$action(next(i_0, see(e))) \tag{1.9}$$
6. Repeat from 2.

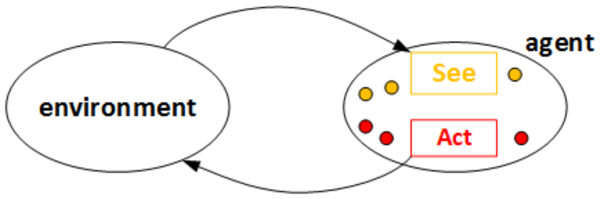


Figure 1.5. Agent Control Loop Diagram

The committee on Visionary Manufacturing Challenges for 2020 has identified IAs as one of the key enabling technologies that will help companies to overcome one of six of their grand challenges and remain productive and profitable in the year 2020. This grand challenge is to instantaneously transform information from a vast array of diverse sources into useful knowledge and effective decisions [31].

Agent-based technologies cannot realize their full potential, and will not become widespread, until standards to support agent interoperability are available and used by agent developers and adequate environments for the development of agent systems are available. Several researchers are working towards the standardization of agent technologies such as the Foundation for Intelligent Physical Agents (FIPA) [32], and in the realization of development environments to build agent systems such as JADE [33].

In addition, Agent Communication Languages (ACLs) are considered as one of the most important specifications of MAS. Agent communication is based on message passing, where agents communicate by formulating and sending individual messages to each other. FIPA-ACL (Foundation for Intelligent Physical Agents - Agent Communication Language) specifies a standard message language by setting out the encoding, semantics, and pragmatics of the messages. The standard does not set out a specific mechanism for the internal transportation of messages. Instead, since different agents might run on different platforms and use different networking technologies, FIPA specifies that the messages transported between platforms should be encoded in a textual form. It is assumed that the agent has some means of transmitting this textual form [34].

Until recently, MAS has been rarely used in energy distribution applications. For example, an ontology design using three tools of Matlab, JADE, and RT-Lab with FIPA-ACL was employed for monitoring voltage regulation in power distribution networks. JADE is a software framework for developing agent-based applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. The goal is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents. JADE can then be considered an agent middleware that implements an agent platform and a development framework. It deals with all those aspects that are not peculiar to the agent internals and that are independent of the applications, such as message transport, encoding and parsing, or agent life cycle. Table 1.3 summarizes some of the specifications of FIPA, JADE, and ACL, which were mentioned previously in this chapter. They are considered the most common standards/applications used for agent design, development, and communication respectively. In section 1.5, FIPA is explained in detail.

Table 1.3. Some of the most common standards/applications used in agent-based systems

Standards/Application	Task in Agent-Based Systems	Specifications
FIPA	Standard for Design	Ontology design, high flexibility, and capability of using three applications: MATLAB, JADE and RT- Lab
JADE	Development environments to build system platforms	Using Java programming code with extra JADE libraries
ACL	Transfer light weight encoded messages between agents	Transfer Java messages but it also converts another message types to Java ACL Message

1.5. Foundation for Intelligent Physical Agents (FIPA) Specifications

FIPA is an international non-profit association of companies and organizations sharing the effort to produce specifications of generic agent technologies. FIPA is envisaged not just as a technology for one application but as generic technologies for different application areas, and not just as independent technologies but as a set of basic technologies that can be integrated by developers to make complex systems with a high degree of interoperability. FIPA is based on two main assumptions. The first is that the time to reach consensus and to complete the standard should not be long, and, mainly, it should not act as a brake on progress rather than an enabler, before industries make commitments. The second is that only the external behavior of system components should be specified, leaving implementation details and internal architectures to agent developers. In fact, the internal architecture of JADE is proprietary even if it complies with the interfaces specified by FIPA [35].

The first output documents of FIPA, called FIPA97 specifications, specify the normative rules that allow a society of agents to inter-operate, operate and be managed.

The specification describes the reference model of an agent platform. It identifies the roles of some key agents necessary for the management of the platform and specifies the agent management content language and ontology. Three key mandatory roles are identified into an agent platform: Agent Management System (AMS), Agent Communication Channel (ACC), and Directory Facilitator (DF). AMS is an agent that exerts supervisory control over access and use of an agent platform; it is responsible for the authentication of resident agents and control of registrations. ACC is an agent that provides the path for basic contact between agents inside and outside the platform. DF is an agent that provides a yellow page service to the agent platform. Note that no restriction is given to the actual technology used for the platform implementation: e-mail based platforms, Common Object Request Broker Architecture (CORBA) based, and Java multi-thread applications could all be FIPA compliant implementations.

FIPA supports common forms of inter-agent conversations through the specification of interaction protocols, which are patterns of messages exchanged by two or more agents. Such protocols include ranges from simple query-request protocols, to the well-known contract net negotiation protocol and English and Dutch auctions. Other parts of the FIPA standard specify other aspects, in particular the agent-software integration, agent mobility and security, ontology service, and the Human-Agent Communication. Agents employ different behaviours to communicate with each other. In the following chapter some of these behaviours are explained.

1.6. Communication among Agents in MAS Using Three Different Behaviours

As Figure 1.6 depicts, architecture of MAS used in power systems consists of three different layers: reactive layer, modeling (coordinator) layer and planning (deliberative) layer [36], [37], [38]. Each agent designed in an MAS is placed in different layers based on its task (e.g., the coordinator layer agent is assigned to the middle layer and applied as a bridge to communication processes [39]. Therefore, each agent used in the power systems could be found in one of the three layers in Figure 1.6.

Agents are located in the reactive layer if they are pre-programmed to do certain tasks. Moreover, agents are placed in the planning layer if they are context dependent, cooperating in their local tasks and competing with similar agents in other nodes in pursuing global goals. In addition, coordinator agents may communicate with agents in the other two layers. Communication among agents allows a cooperative-based approach to system control in microgrids. This enables utilization of an MAS approach to converter coordination of different power devices including energy storage.

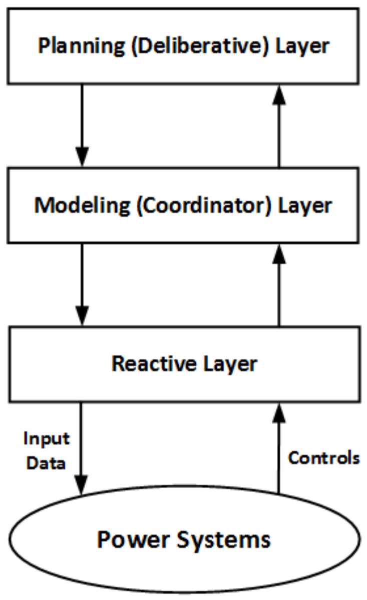


Figure 1.6. Three Layers of MASs

In MAS, each agent is capable of different behaviours based on design requirements such as brokerage, facilitating, and matchmaking [40]. As Figure 1.7 depicts, broker behavior manages the sharing process by receiving information from the other agents and allocating the protocols for agents to communicate with the brokers [41]. Agent1 as Broker Agent (BrAg) initiates communication with all the other agents to detect the lowest cost path. In the next step, all of the other three agents propose their costs. Upon choosing the compatible agent, Agent1 informs all the other agents. Facilitator Agent (FacAg) communicates with all the agents to establish the path. It detects the suitable agents for running any task. In addition, it uses a Matchmaker Agent (MchAg) to establish collaborative sub-systems of agents.

A matchmaker agent pairs two agents of service provider and service requester for matching given requests of requester agents with appropriate advertised services of registered provider agents. In contrast to the functionality of both the broker and facilitator, it simply returns a ranked list of relevant provider agents to the requesting agent. Consequently, the requester agent has to contact and negotiate with the relevant provider agent itself for getting the services it desires. This direct interaction between requester and selected provider agents is performed independently from the matchmaker. It avoids, for example, data transmission bottlenecks or single point of failure at the matchmaker but increases direct communication overhead between matched requester/provider agents [42].

Therefore, a collaborative subsystem of agents is established using the MchAg. The individual agents start to communicate directly with each other through established links and data do not need to pass through MchAg anymore.

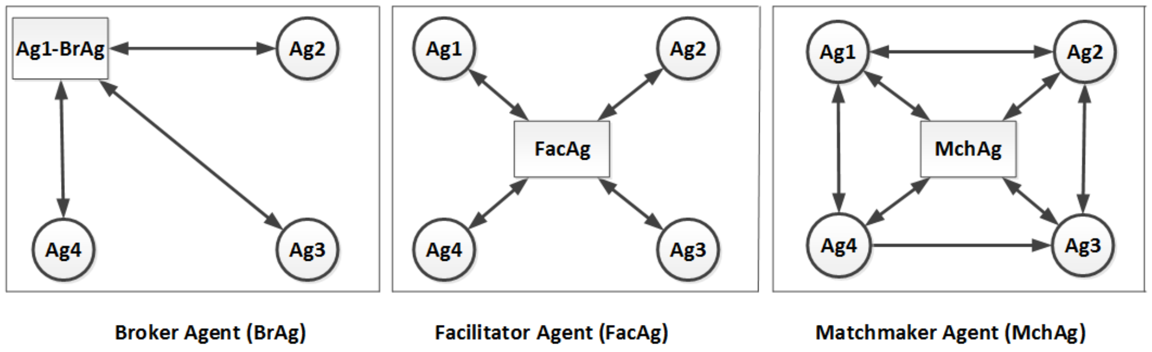


Figure 1.7. Communication among Agents Using Three Different Behaviours (Broker, Facilitator, and Matchmaker)

Referring to the categorizing real-time control agent technologies in the Brennan paper [43], it is realized that the facilitator and MchAgs are the most appropriate options for the sample power system. The matchmaking algorithm is used to establish links among agents and delete the MchAg after establishment. A facilitator agent also is required to provide reliable network communication among agents. In other words, the broker technology doesn't seem to be a good choice for real-time middle-sized microgrids because all the messages have to communicate with one single source or server, and it decreases the communication time. Moreover, in some networks such configuration may not offer adequate scalability. Furthermore, it does not provide a satisfactory level of fault-tolerance since crashing a single broker may result in a large number of state transfer operations during recovery. Also there is another agent technology that is called publish-subscribe which includes a combination of facilitating and matchmaking behaviours. In the chapter 3 the efficiency of aforementioned agent technologies using defined metric (number of message exchanges0metricvalue for a test system.

1.7. Thesis Structure including the Application of Three Microgrids as Case Studies and their Associated Agent Control Models

In this thesis, consideration will be given to three types of microgrids as case studies; one of them is a medium-sized islanded DC microgrid including 32 converters. It is used for numerical analysis and selection of the most efficient coordinator control for islanded DC microgrids and is called MicroGrid-32 (MG-32). The second case study including eight converters called MicroGrid-8(MG-8) which is a smaller size of the same type of microgrid. It is used as a case study for evaluating proposed agent-based control platform, modeled in the MATLAB. The last test system used in this research is a hierarchical agent-based model, which is developed and employed for Volt-VAR Optimization (VVO) in a small-sized connected AC microgrid. The system is including 16 power devices and it is called MicroGrid-16 (MG-16).

As explained in section 1.6, each agent designed in an MAS is placed in different agent layers of planning, coordinator, and reactive based on its tasks. When real-time tasks are required, they are usually handled using reactive layer's agent because they are closely associated to power electronic devices such as converters [44]. This thesis proposes an MAS to coordinate distributed power generators, optimize real-time, agent-based algorithms, and achieve higher degrees of efficiency and reliability. In this thesis two different agent technologies are used in microgrids including: Volt-Var Optimization (VVO) for a grid-connected AC microgrid and coordination among power converters for an islanded DC microgrid. Their capabilities are compared based on each microgrid's requirements:

- VVO model is defined on a broad timing range of 1-15 minutes while an islanded DC microgrid is considered as soft real-time systems with a deterministic time frame of a few tens of milliseconds.
- VVO is a small-sized microgrid including eight smart meters and a couple of other power devices like capacitor banks, but DC microgrid is a medium-sized microgrid including tens of converters.
- Optimization is applied in all three agent layers of planning, coordinator, and reactive layer for the VVO test system, while the islanded DC microgrid deployed coordinator and reactive layers.

For the first test system used in this research, an agent-based control system for voltage regulation among converters in a defined DC microgrid. Two different agent platforms are developed in this thesis. Considering that the islanded DC microgrid has a low inertia, the designed agent platform needs to perform a very fast convergence among power electronic devices. In this research, agent technology is applied for control action in middle-sized DC microgrids as one of the most efficient and scalable solutions [45].

The second test system is on the design and development of a hierarchical agent-based model is developed and employed for Volt-VAR Optimization (VVO) in an AC microgrid. VVO is one of the main techniques which is traditionally applied in power systems to reduce losses in distribution feeders [46]. It is an advanced method that optimizes voltage and/or reactive power (VAR) of a distribution network based on a predetermined aggregated feeder load profile. Due to the effects of load profiles on the quality of delivered energy to the customers, effective technologies need to be developed to monitor and control the amount of reactive power and energy losses in distribution systems. This research proposes a real-time adaptive approach for an integrated VVO in a smart grid applying a distributed command and control platform with IAs. It presents a primary MAS topology and structure with respect to the VVO application. In chapter 6, the design and implementation procedures of VVO agent technology is presented.

Table 1.4 compares some of the specifications of VVO and DC shipboard microgrid which lead to the selection of different agent technologies for these two test systems. Figure 1.7 categorized MAS into three types of broker, facilitator, and matchmaker based on communication among agents in an MAS based on their behaviour. While control requirements for the VVO test system are addressed using facilitator agent design, control requirements for the DC Shipboard microgrid are provided deploying matchmaker agent technology. Table 1.5 describes the tasks and naming of three case studies that are grouped in two test systems.

Table 1.4. Comparison between two test systems: VVO and DC Shipboard microgrid (DG: Distributed Generators, LTC: Load Tab Changer, CB: Capacitor Bank)

Features of Test Systems	VVO	DC Shipboard Microgrid
Size of System	Small-sized with less than 16 power devices including 8 smart meters, 2 CBs, and 2 LTC	Medium-sized with tens of converters connected to the all power devices including DGs
Timing	Broad with a cycle of running system within one minute for agents in reactive layer and 15 minutes for agents in planning layer	Soft real-time with a constrain on having deterministic time frame of a few tens of millisecond
Associated Agent Layer	Planning, Coordinator, and Reactive	Coordinator, and Reactive
Applied Agent Technology	Facilitator	Publish-Subscribe

Table 1.5. Description of Test Systems and Case Studies in Thesis

Test System - Case Study	Microgrid Model	Associated Agent Platform	System Task
1-1	MicroGrid-32 (MG-32) Medium-sized islanded DC	Real-time Coordinator of Power Converters with 32 Agents (RCPC-32)	Used for numerical analysis and selection of the most efficient coordinator control for islanded DC microgrids
1-2	MicroGrid-8 (MG-8) Small-sized islanded DC	Real-time Coordinator of Power Converters with 8 Agents (RCPC-8)	Used for evaluating proposed agent-based control platform
2-1	Microgrid-16 (MG-16) Small-sized connected AC	Volt-VAR Optimization (VVO)	Developed and employed for Volt-VAR Optimization (VVO)

Chapter 2. Evaluating Commonly used Distributed Agent Control Technologies for Microgrids (First Case Study)

Agent-based technologies are still immature, and few truly agent-based systems have been realized [47]. Agent-based technologies cannot realize their full potential, and will not become widespread, until standards to support agent interoperability become available and used by agent developers and until adequate environments for the development of agent systems are developed. Several researchers are working towards the standardization of agent technologies (see for example, the work by Knowledge Sharing Effort [48], OMG [49] and FIPA [50]) and in the realization of development environments to build agent systems (see, for example, RETSINA [51]).

However, the use of a common communication language is not enough to easily support interoperability between different agent systems. The work of FIPA, as explained in section 1.5, is in the direction to allow an easy interoperability between agent systems. FIPA’s task can be summarized as follows: 1) FIPA standardizes agent communication language (ACC); 2) It specifies the key agents necessary for the management of an agent system (AMS); 3) It declares the required ontology for the interaction between systems (DF); 4) It defines the transport layer of the protocols. In this thesis, JADE is used as a software framework to develop agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems.

A Real-time Coordinator of Power Converters (RCPC) system is defined as a control model for the sample microgrid. To test the applicability of the RCPC optimization method, an MG-32 model is used as a case study for numerical analysis, which is called RCPC-32. In chapter 3, where a simplified case study including eight converters (MG-8) is used for simulation, its associated agent-based control system is called RCPC-8, which is basically a smaller size of RCPC-32. Different control designs are employed in an MG-32 system and their efficiency and scalability is compared. A detail explanation about aforementioned systems is included in Appendix B. In this section, RCPC-32 is designed as an agent-based system to combine the predictability of the centralized and hierarchical control architectures with the agility and robustness against disturbances and high degree of adaptability of the heterarchical control architectures [52].

Figure 2.1 illustrates the application of agent technology in an RCPC-32 system. Each Converter Agent (CA) is connected to one of 32 converters using Ethernet communication protocols (TCP/IP or UDP/IP) located in the lower layer. A group of CAs is assigned to each middle layer agents. The number of Middle Agents (MAs) and the topology of their connection to both CAs at the lower layer and Planning Agent (PA) which also known as MchAg at the upper layer is varied for different agent algorithms. There is only one PA that is responsible for saving and mapping system plans.

In the following sections, a few of the most efficient algorithms extracted from literature are reviewed and two of them customized for an RCPC-32 system (Figure 2.1). In addition, they are clarified using a case study, and their efficiencies are compared based on defined metrics. The aforementioned algorithms include:

1. Belief, Desire, Intention (BDI) Architecture, using bidding algorithms; Designed for Game Theory and simulated Robotic Arms [53].
2. Holonic Control Architecture (HCA); Designed for global data centric such as internet distributed networks [54], [55].
3. Intelligent Distributed Autonomous Power System (IDAPS); Designed for Smart Grids.
4. A combination of publish-subscribe and a decentralized synchronization mechanism including Distributed Hash Table (DHT) and Consistent Hash Table (CHT); Applied for different distributed systems.

The two latter ones use FIPA agent design standard. In the following four sections, each of the aforementioned optimization algorithms are explained and the first and the last one are clarified using the case study and their efficiencies are compared using defined metrics. Based on the research hypothesis, the selected optimization algorithm should maintain a deterministic time frame of a few tens of milliseconds for a system with tens of converters with no limitation in the number of events which might happen concurrently.

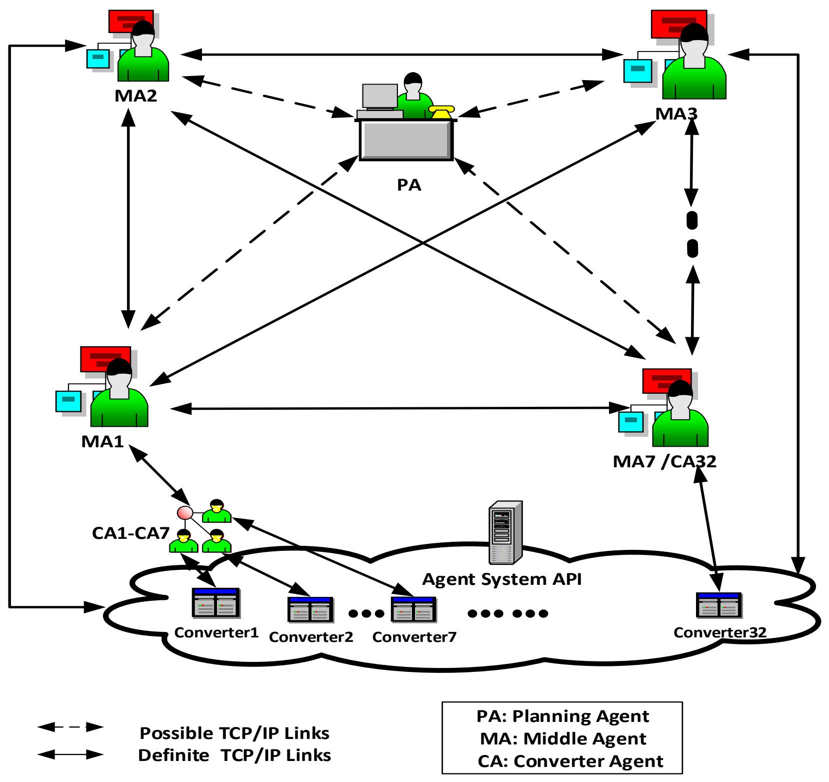


Figure 2.1. Communication Design between Three Types of Agents in RCPC-32 System Applied for Coordinator Control of Converters in MG-32

2.1. Belief, Desire, Intention (BDI) Architecture over Bidding Algorithm

This algorithm is mainly designed for game theory and simulated robotic arms, and includes three main parts: belief, desire, and intention. It has been developed based on hierarchy architecture and customized for an RCPC-32 system.

- **Beliefs**: current information of agents that consists of:
    - Perceiving the Environment—for example:
        - Receives terms of current decomposition from Power Electronic Building Block (PEBB), which are defined as power converters

        - Analyzes operating conditions to detect violations of power quality requirements [56]
    - Itself:
        - Apparent power limit
        - Current utilization (as computed from current measurement)
        - Ability to contribute to correction of imbalance load (based on energy storage availability)
        - Ability to contribute to active power compensation
    - Peer agent: Manages sharing process by receiving information from other agents and allocating compensation among them.
- **Desires**: set long-term goals that agents want to achieve, such as:
    - Effective and flexible management of energy flow throughout microgrids.
    - Distributed implementation to avoid single points of failure that makes it robust and expandable.
        - Acts as higher level controller of its assigned converter, feeding the compensation parameters to the controller.
    - Agent immediate commitment to handle events by looking for plans that match the events and lead to desires.
- **Intentions**: make a plan and become committed to its intention.
    - As it executes a plan, it may generate new events that require handling.

2.1.1. Bidding Algorithm

Figure 2.2 compares the results of the three bidding algorithms computed with similar data values using the cost formula (3.1), (3.2). It is clear that the sequential algorithm has the least complexity and therefore is the most efficient and persistent approach. Computational weight for each iteration is O(mn), where n and m are the numbers of agents and tasks, respectively. Considering the worst case scenario by assuming that all of tasks are being run in parallel, then m=n. Therefore, the bidding cost can be calculated from (3-1) to (3-3) for simultaneous, sequential, and dynamic bidding algorithms respectively. System requirements for the defined microgrid requires running concurrent events on systems' agents, so simultaneous algorithms are chosen as the preferred method for this research. The computational weight for each iteration is O(n*m) where n is the number of the auctioneer and m is the number of bidders. Considering the worst case scenario, when all the agents are bidding concurrently, the bidding cost is calculated from (3.1).

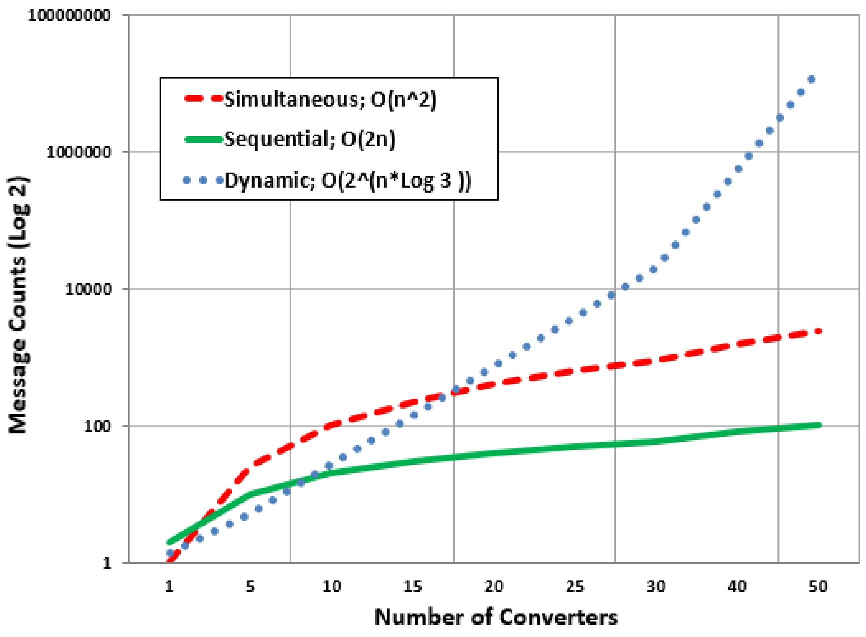


Figure 2.2. Comparison among Bidding Lookup Algorithms Based on Complexity

For Simultaneous Bidding Algorithm:

$$Cost \leq (n * m) \quad \& \quad m = n \Rightarrow \; Cost \leq (n * n) = n^2 \qquad (3.1)$$
$$where\; 0 \leq n \leq 32$$

For Sequential Bidding Algorithm:

$$Cost \leq (n + m) \quad \& \; m = n \; \Rightarrow \; Cost \leq (n + n) = 2n \qquad (3.2)$$
$$where\; 0 \leq n \leq 32$$

For Dynamic Bidding Algorithm:

$$Cost \leq \left(2^{n \log 3}\right) \; \Rightarrow \; Cost \leq (2^{n \times 1.58}) \qquad (3.3)$$
$$where\; 0 \leq n \leq 32$$

2.1.2. Case Study for BDI Algorithms

As mentioned before, a microgrid including 32 power converters is considered as a testbed for the design of BDI control algorithms. Figure 2.3 illustrates the agent management schema for the case study.

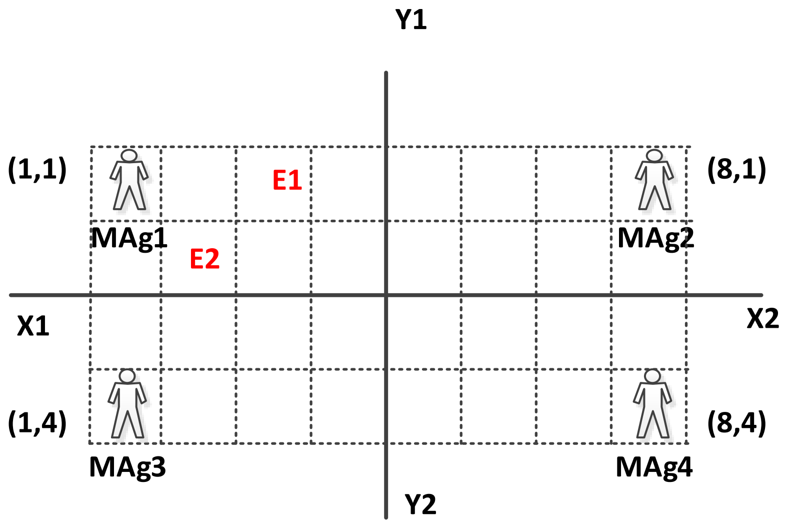


Figure 2.3. Agent Management Schema

Environment: Four MAgs equally coordinate 32 CAgs and each MAg is connected to a power converter [57]. Each agent can directly handle its local events or other events received from adjacent agents (contest simulator). Figure 2.4 demonstrates the transaction between MAg and its environment.

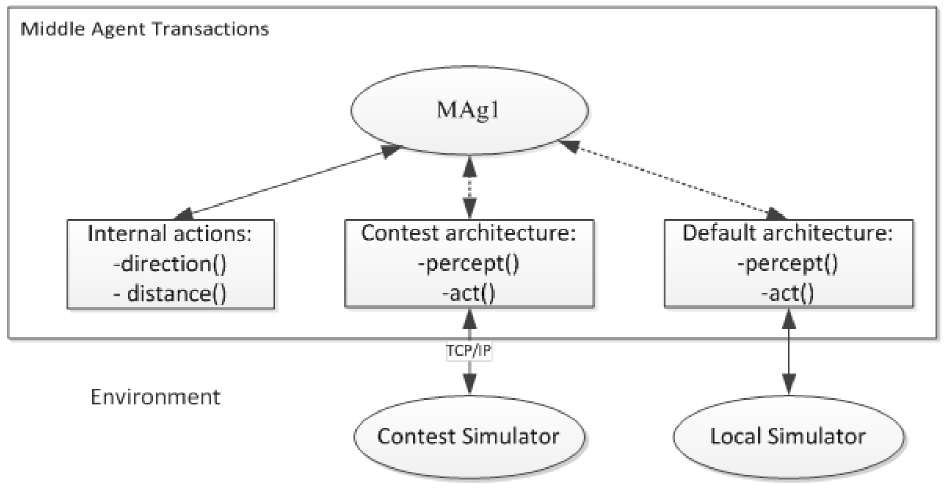

Middle Agent Transactions
MAg1
Internal actions: -direction() - distance()
Contest architecture: -percept() -act()
Default architecture: -percept() -act()
TCP/IP
Environment
Contest Simulator
Local Simulator

Figure 2.4. Transaction among a Middle Agent and Its Environment

The following scenario is defined in communication among peer MAs (Figure 2.5):

1. MA1 allocated at (1, 1).
2. MA1 detects E1 (event1) at (3, 1).
3. MA1 goes to (3, 1).
4. MA1 announces to PA that E1 is at (3,1).
5. MA1 announces to MA2, MA3, and MA4 its situation.
6. MA1 detects E2 at (2, 2).
7. MA2, MA3, and MA4 bid for handling event.
8. MA1 reports E2 (2, 2) to MA3 (which is the closest agent to the area).
9. MA3 wins the bid (because it is the closest agent to the area and has lowest cost)
10. MA3 goes to (2, 2).
11. MA3 announces its current situation to MA1, MA2, and MA4.

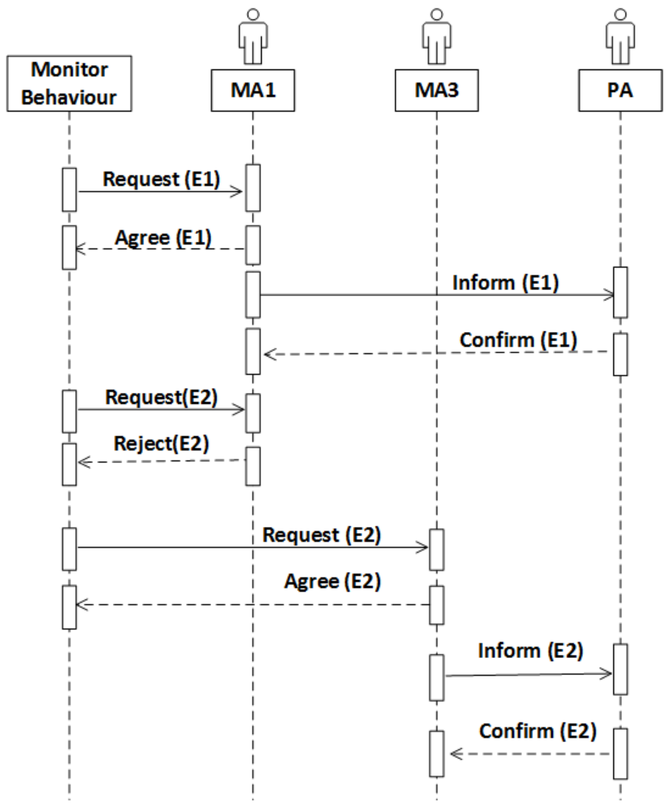


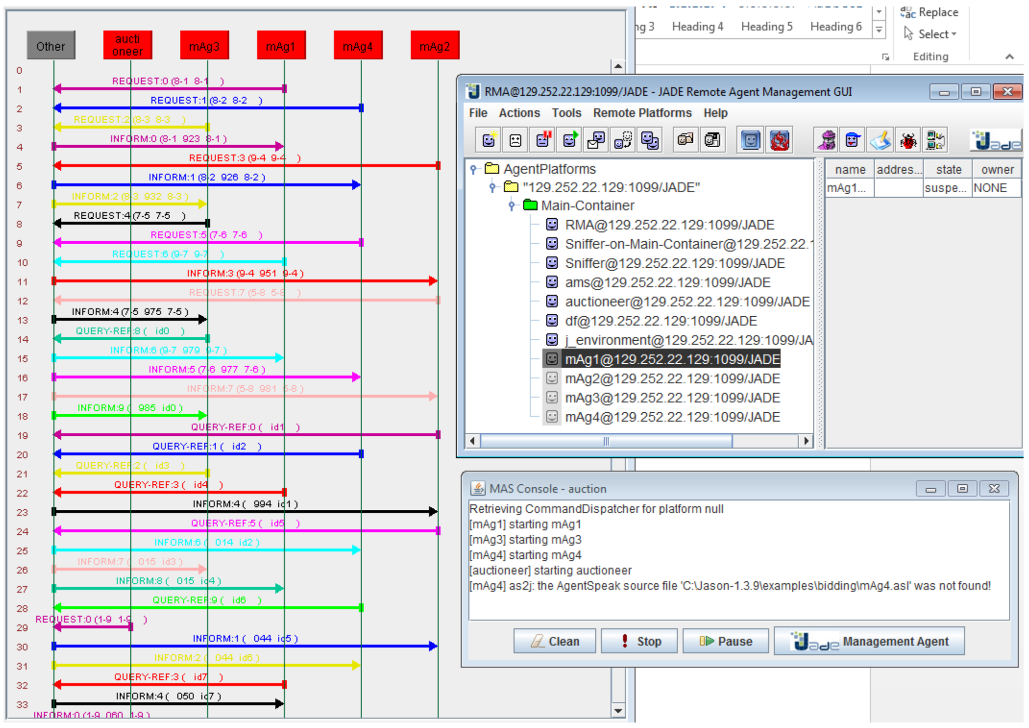
Figure 2.5. Communication among Peer Agents

Jason IDE and JADE agent programming language are used to define and program MAgs with their associated IP addresses as their unique (Agent IDs) AIDs. They are able to communicate with each other and exchange data. The canvas on the right of Figure 2.6 provides a graphical representation of the messages exchanged between sniffed agents, where each arrow represents a message and each color identifies a conversation. When the user decides to sniff an agent or a group of agents, every message directed to, or coming from, that agent/group is tracked and displayed in the sniffer GUI. Each of the colors stand for one type of message:

- Purple: initialing data transfer and data from initiator agent (e.g., MAg);
- Blue: exchanging data among running agents;
- Yellow: sending final data value to initiator agent (e.g., MAg);

- Black: stopping agents.

Some of the most commonly used message types are inform, request, agree, not understood, and reject. They capture the essence of most forms of basic communication. As all the agents are installed on home IP Address and communicate through the same port, their socket numbers are an identical number of :129.252.22.129:1099. Port number 1099 is associated for agents by networking standards, so each time that an agent-based program runs, it gets occupied and should be cleared before running the next program.



AgentPlatforms
"129.252.22.129:1099/JADE"
Main-Container
RMA@129.252.22.129:1099/JADE
Sniffer-on-Main-Container@129.252.22.1
Sniffer@129.252.22.129:1099/JADE
ams@129.252.22.129:1099/JADE
auctioneer@129.252.22.129:1099/JADE
df@129.252.22.129:1099/JADE
j_environment@129.252.22.129:1099/JA
mAg1@129.252.22.129:1099/JADE
mAg2@129.252.22.129:1099/JADE
mAg3@129.252.22.129:1099/JADE
mAg4@129.252.22.129:1099/JADE

name	addres...	state	owner
mAg1...		suspe...	NONE

```
Retrieving CommandDispatcher for platform null
[mAg1] starting mAg1
[mAg3] starting mAg3
[mAg4] starting mAg4
[auctioneer] starting auctioneer
[mAg4] as2j: the AgentSpeak source file 'C:\Jason-1.3.9\examples\bidding\mAg4.asl' was not found!
```

Figure 2.6. Communication among Middle Agents Using a JADE Sniffer Tool in a Jason Application

2.1.3. Advantages and Disadvantages of BDI

Here are the advantages and disadvantages:

Advantages:

- Bidding procedure runs for four MAgs in less than 1.25 seconds for an MG-32 system;

- A Jason platform is already used in the implementation of MAS as a successful application.

Disadvantages:

- The bidding algorithm is used for choosing the winner agent; therefore, the system speed decreases exponentially by increasing the number of MAs (Figure 2.2), so it is not efficient for middle and large size microgrids.

2.2. Holonic Control Architecture (HCA)

In a hierarchical architecture, there are multiple layers of master/slave agent relationships where agents at one layer of the hierarchy are slaves to a master agent at the next highest layer of the hierarchy. In the Holonic manufacturing community, a hierarchy of agents (holons) is called a holarchy58 [59]. A holon is distinguished from a software agent by adding a physical processing part, such as a converter, to an information processing part, which is an MAS in this thesis [60]. The IEC 61499 standard can be used for functional blocks in a distributed automation of a Holonic design [61]. Since it uses a Low Level Control (LLC), the system response time is between 10–100 milliseconds, which is an acceptable time for running an RCPC-32 system. Figure 2.7 displays a simple schema of a Holonic control system.

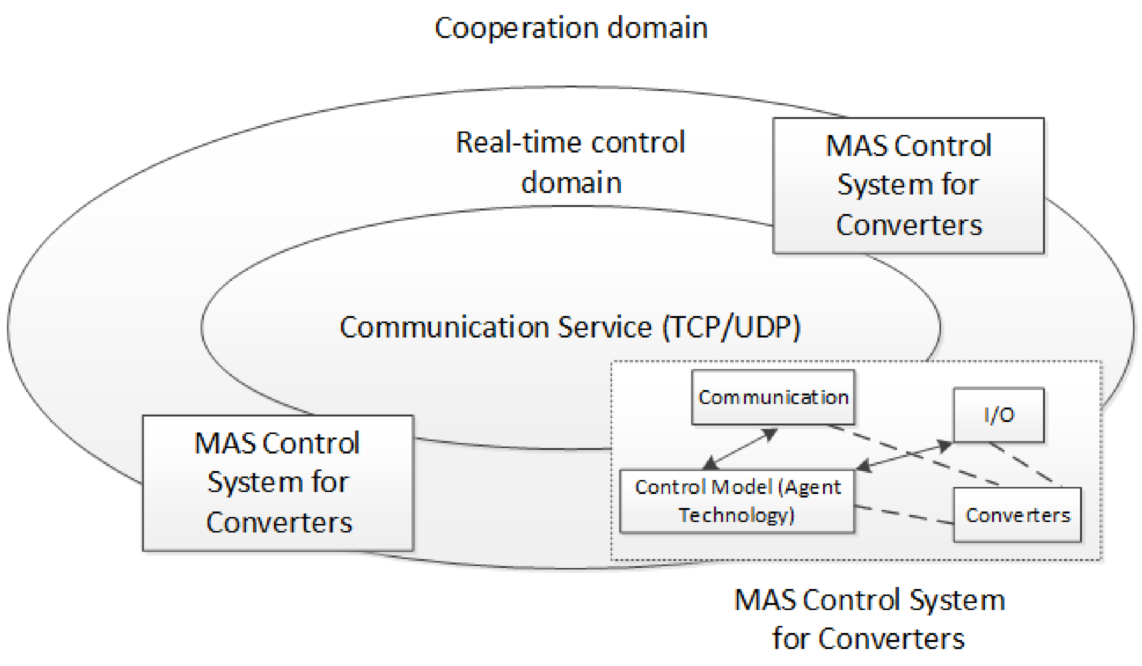

Cooperation domain
Real-time control domain
MAS Control System for Converters
Communication Service (TCP/UDP)
Communication
I/O
Control Model (Agent Technology)
Converters
MAS Control System for Converters
MAS Control System for Converters

Figure 2.7. Holonic Low-level Control Architecture for an RCPC-32 System

2.2.1. Advantages and Disadvantages of HCA

Advantages:

- The processing time is between 10–100 milliseconds for each event (e.g., changing the system load);
- The hardware and software connection is defined based on an IEC 61499 protocol, which is an international standard for distributed systems

Disadvantages:

- It only works with software and hardware that meet the IEC 61499 standard;

- Some of the most common lookup algorithms, such as the bullying algorithm, which is required for MAS, have never been tested for Holonic control architecture;

- Agents are designed to be coded using JADE, but any special platform is recommended to be compatible with other algorithms and an IEC 61499 protocol.

2.3. Intelligent Distributed Autonomous Power System (IDAPS)

IDAPS has been used as a potential backup for system protection in microgrids. It properly works in the island mode. By design, the IDAPS is a specialized microgrid that is set up at the level of end-use customers to allow electricity trading among neighbors [62]. During a power outage, an IDAPS cell will island itself from the grid and start to operate autonomously. Critical loads will first be served by their internal sources, and any shortfall can be made up through open market purchases. It is also possible to buy and sell electricity among different IDAPS microgrids [63]. As Figure 2.8 displays, the main operation characteristics of IDAPS are: supply-driven demand, multi-agent technology, and web-based communication architecture.

In IDAPS, a multi-agent system was developed using JADE, a microgrid was simulated in Matlab, and a TCP/IP server was implemented to allow a single TCP/IP connection to an external system at a time. All agent actions—from detecting the fault, disconnecting the main circuit breaker, disconnecting the noncritical loads to stabilizing the grid—can be accomplished within half an electrical cycle (i.e., less than 0.008 seconds for a 60-Hz system); that is one of the biggest advantages of applying this system for protection.

Advantages:

- Taking less than 10 milliseconds for processing a task in a 60-Hz power system;
- Implementing an integrated system for hardware, software and user interface.

Disadvantages:

- Only one single TCP/IP connection was allowed to an external system at a time. As a result, a middle server was developed later on to allow multiple TCP connections to the external MAS. But this server can only get multiple commands from MAS to transfer to microgrid. Thus, it would not be able to address more than one event at the same time.

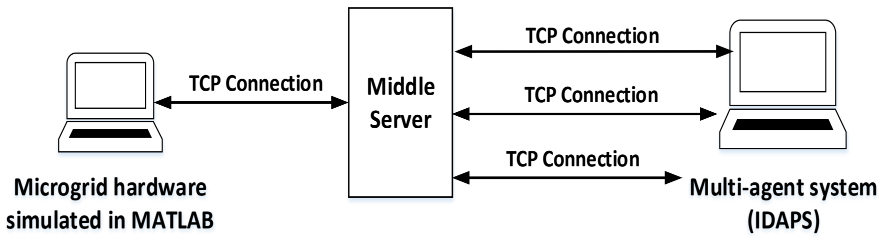


Figure 2.8. IDAPS System Total View

2.4. Publish-Subscribe Using DHT

Publish-subscribe is a messaging pattern in which message senders (publishers) do not program the messages to be sent directly to specific receivers (subscribers). Published messages are categorized into classes. The subscribers express interest in one or more classes, and only receive messages that are of interest. This is a novel method of task allocation among multi-agent systems. The unique concept of publish-subscribe systems is that messages are addressed by content rather than by destination. This idea, often called subject-based addressing, is used to divide the network into a loosely coupled association of anonymous data producers and data consumers [64]. As a compatible algorithm, hash tables have been used for efficient message routing among agents.

In the publish-subscribe algorithm, the following metrics have been identified to measure the performance: latency, total number of messages generated for Resource Lookup Queries (RLQs), response time, and routing hops. As seen, message count is the most common metric value among the parameters; thus, it has been chosen as a unique metric value for evaluating algorithms in an RCPC-32 system. The measurement parameters, such as RLQs and routing hops, are averaged over all the broker service in the system. This process of routing the event to the peer responsible for managing it takes

$O(dN^{1/d})$ overlay hops on average, where N is the number of peers in the system and d is the dimensionality of the cartesian space [65].

Publish-subscribe systems are becoming increasingly popular in building large distributed information systems. In such systems as shown in Figure 2.9. A, subscribers specify their interests to the system using a set of subscriptions [66]. Publishers submit new information into the system using a set of publications. Upon receiving a publication, the system searches for matching subscriptions and notifies the interested subscribers. Unlike the client/server model, the publish-subscribe model decouples time (Figure 2.9. B), space, and flow between publishers and subscribers, which may lead to benefits such as reduced program complexity and resource consumption.

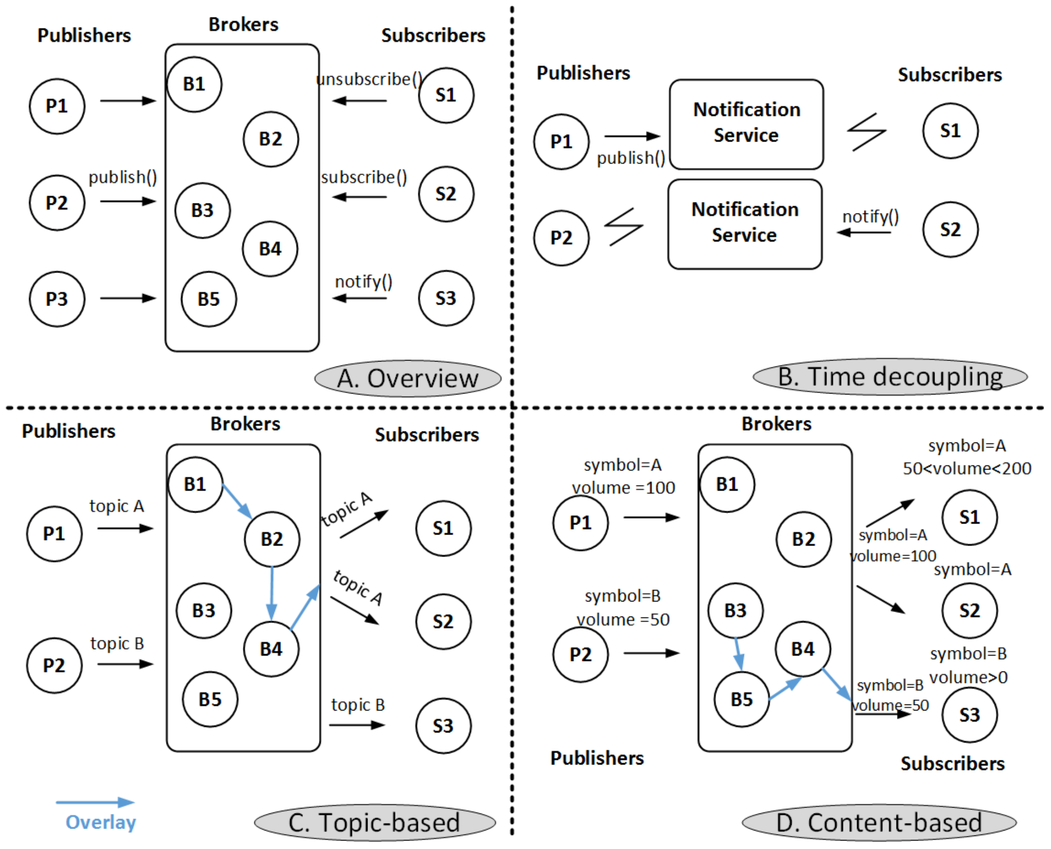


Figure 2.9. Overview of Publish-Subscribe on Broker Overlay

There are at least two major classes of publish/subscribe systems which are shown in Figure 2.9.C&D:

1. Topic-based subscribers join a group containing a topic of interest. Publications that belong to the topic are broadcast to all members of the group. Therefore, publishers and subscribers must explicitly specify the group they wish to join. Topic-based systems are similar to the earlier group communication and event-notification systems (e.g., in newsgroups).
2. In content-based publish/subscribe systems, the matching of subscriptions and publications is based on content and no prior knowledge is needed (e.g., the set of available topics). Therefore, these systems are more flexible and useful since subscribers can specify their interests more accurately using a set of predicates [67], [68].

However, the main difficulty in building distributed content-based systems is the design of an efficient distributed matching algorithm. Existing distributed content-based systems typically rely on a small number of trusted brokers that are inter-connected using a high-bandwidth network (Figure 2.9) [69]. In some scenarios, such configurations may not offer adequate scalability. As well, they do not provide a satisfactory level of fault-tolerance since crashing a single broker may result in a large number of state transfer operations during recovery.

Content-based systems support a wide set of queries including prefix, suffix, containment, equality predicates on strings, and range and comparison predicates on numerical-typed attributes. P2P data networks may be very useful to build queries using the publish-subscribe paradigm. A comprehensive infrastructure is contributed which supports efficiently and scalability a rich set of operators on string and numerical-typed attributes. The matching of publications (a.k.a. events) to subscriptions (a.k.a. interests) is done based on the content (values of attributes). In addition, in content-based model , the publisher is given the ability to express their interest by specifying a period of defined values over different attributes. Thus, the subscribers of content-based publish-subscribes are made of {attribute, operator, value} tuples, where the operator can be one of {<, =, >, ≤, ≥}, and the publishers are a set of {attribute, value} pairs [70], [71].

Recently, DHTs have emerged as an infrastructure for efficient, scalable resource lookup in large peer-to-peer distributed networks. Such systems are decentralized, scalable, and self-organizing (i.e., often, as well, they automatically adapt to the arrival, departure, and failure of nodes in the network) [72]. Such characteristics make DHTs attractive for building distributed applications. In fact, DHTs have successfully been used in several application domains, such as distributed file systems. An overlay network is a network which is made on the top of an existing network. They are a good solution for supporting distributed algorithms such as publish-subscribe systems. DHT is an overlay network which is designed on an application layer of a TCP/IP internet network.

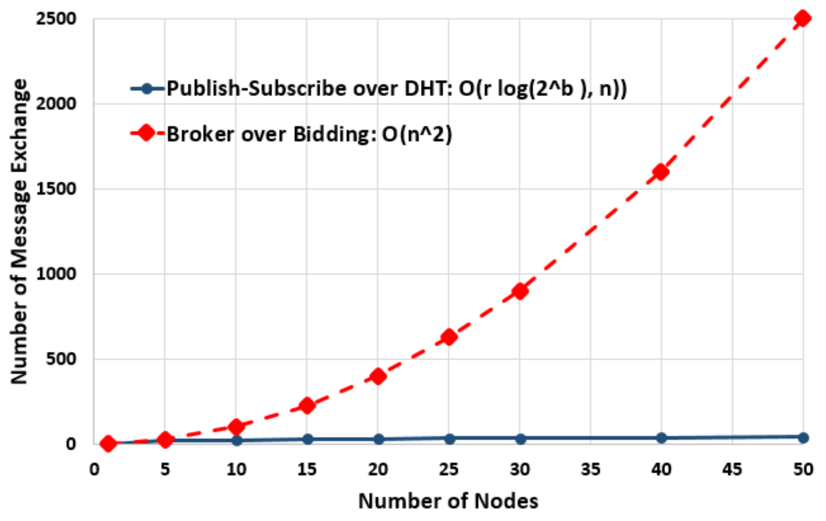
The above method uses a hash table functionality to manage, join, and leave the nodes in a wide-area environment. The sets of (key, value), which are defined in hash tables, help users to retrieve a value corresponding to a given key. The application of publish-subscribe over DHT in large-scale distributed systems has been studied by several authors. It might be argued that in small-scale systems, publish-subscribe can be implemented on a centralized overlay network protocol with lower cost. However, DHT does not require any changes in system architecture for adding the nodes to the system over time [73]. Therefore, the ability to scale up the system size easily and with low cost will retrieve the usage of DHT as a system infrastructure [74].

Ratnasamy, and Francis [75] presented an approach that is able to support string-attribute predicates with message complexity $O(r \times \log_{2^b} n)$ (where r is the average length of string values, 2^b is a configuration parameter with typical value of 2, and n is the number of nodes) for publishers and $O(\log_{2^b} n)$ for subscriptions. The size of the populated portion of the routing table (approximately $\left[\log_{2^b} n\right] \times \left(2^b - 1\right)$ entries) and the maximum number of hops required to route between any pair of nodes ($\left[\log_{2^b} n\right]$). For instance, with a value of b=4 and n=10^6, a routing table contains on average 75 entries, and the expected number of routing hops is 5, whilst with 10^9 nodes, the routing table contains on average 105 entries, and the expected number of routing hops in 7.

As briefly mentioned in section 2.1, considering 32 nodes and one prefix for each member of the routing table, the expected number of rooting hopes will be equal:

$\lceil \log_{2^b} n \rceil = \lceil \log_{2^1} 2^5 \rceil = 5$ for content-based rooting of publish-subscribe algorithm. The routing table size calculates from: $\lceil \log_{2^b} n \rceil \ \times \ (2^b - 1) = \lceil \log_{2^1} 32 \rceil \times (2^1 - 1) = 5 \ \times 1.$

Based on section 2.1.1, the number of message counts for a simultaneous bidding algorithm in a brokerage messaging pattern is calculated by n^2, which grows faster than a publish-subscribe method by increasing the number of nodes. A computational comparison between agent technologies shows that the publish-subscribe mechanism displays consistent distributed control in the upper time limit for a microgrid with a variation from 1 to 50 converters. Figure 2.10 illustrates that a variable number of converters between 1 and 50 affects the number of message exchanges in a publish-subscribe model slightly compared to one of the most used agent technologies known as bidding algorithm. Section 2.6 explains the design of a case study system based on a publish-subscribe model, followed by the application of DHT as an infrastructure in this case study.



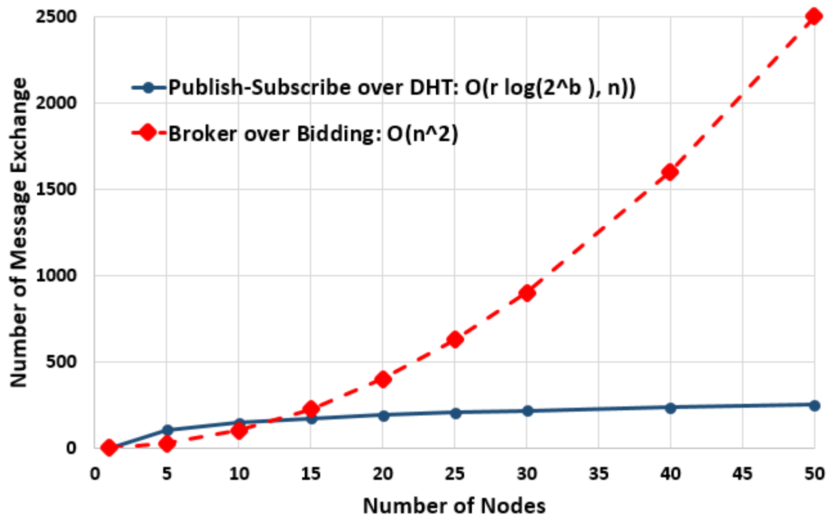


Figure 2.10. A Comparison between Complexity of Bidding and Publish-subscribe Agent Designs Considering Maximum Length of Messages (r=7, and r=44)

2.5. Searching Algorithm Based on DHT for First Case Study

Any DHT could be utilized for routing of n-dimensional index. In the Chord method that is the most popular DHT structure, the complexity for routing table, lookup, and peer joint/leave are O(log n), O(log n), and O($(\log n)^2$) respectively [76]. The main idea of DHT is partitioning tables. Each node gets an identity by hashing its IP address, and keys are also hashed into the same space (Figure 2.11). A key(k) with a hashed ID *k* is assigned to the first node whose hashed ID is equal to or follows k in a circular space:

Successor (k), Put (key, data) → Lookup (key) → data

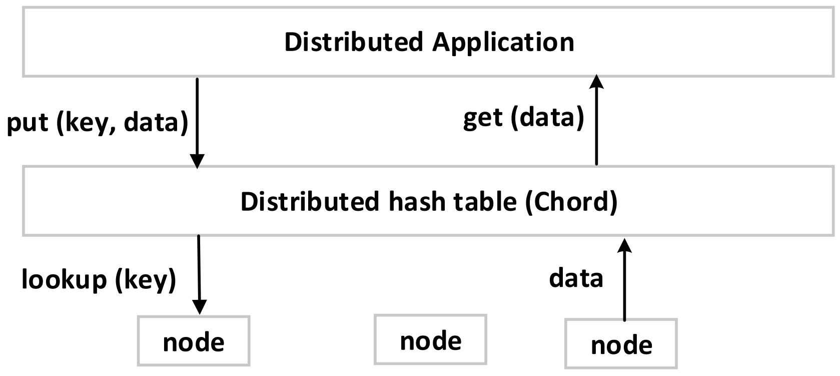


Figure 2.11. Put/get Interface of DHT

In addition to a faster searching speed, this algorithm has some other advantages compared with the same lookup algorithms, previously reviewed in this document, e.g., good load balancing, high efficiency (O(log n) messages per lookup), and robustness. In this algorithm, choosing a list of successors for each node depends on the number of agents as well as the ability of self-healing in the case of failing half of the converters at the same time. Figure 2.12 displays the process of assigning 32 converters to seven MAs.

1. Thirty-two converters are installed in two buses.
2. All of the converters have the capability to become an MA.
3. A variable number of converters are assigned to each MA based on the converters' load value and event and history of event happening history. The MchAg has planned this information.
4. A different number of converters might be working within each IP address range every time.
5. Each added converter will find its place using lookup algorithms in DHT.
6. Each MA has information of its three successors, which leads to ability of self-healing even after failure of half of the working converters.

Only one shortcut in each search is allowed. Figure 2.13 illustrates detailed steps for searching for a destination converter using DHT, while the flowchart of the look up process is displayed in Figure 2.14. The other processes, such as managing load balance in case of adding and deleting a converter, have also been studied, and it is realized that they have similar flowcharts. As observed, DHT and publish-subscribe are considered to be the most efficient methods for implementing an RCPC-32 system [77].

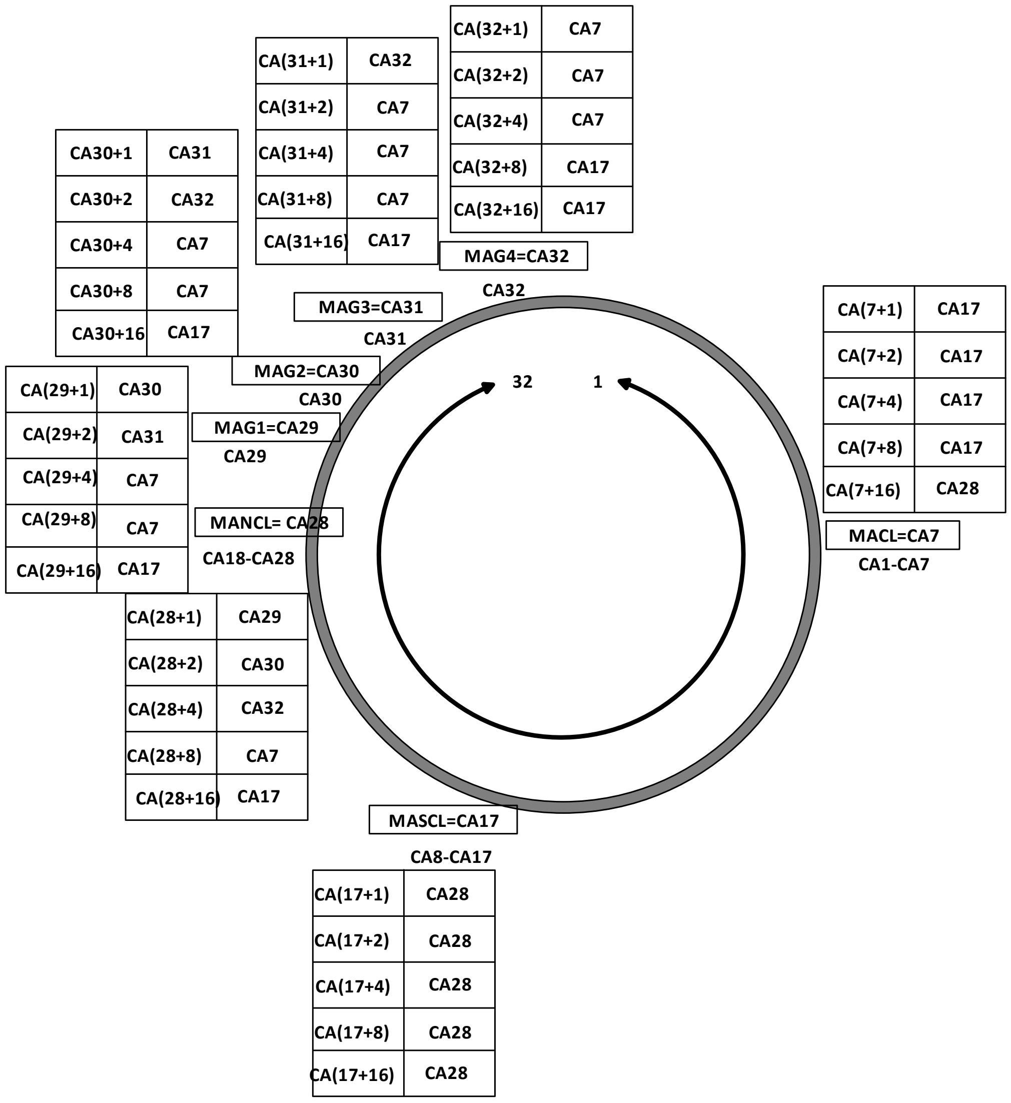


Figure 2.12. Assigning CAs to MAs using DHT for 32 Converters and 7 MAs (CA: Converter Agent, MA: Middle Agent)

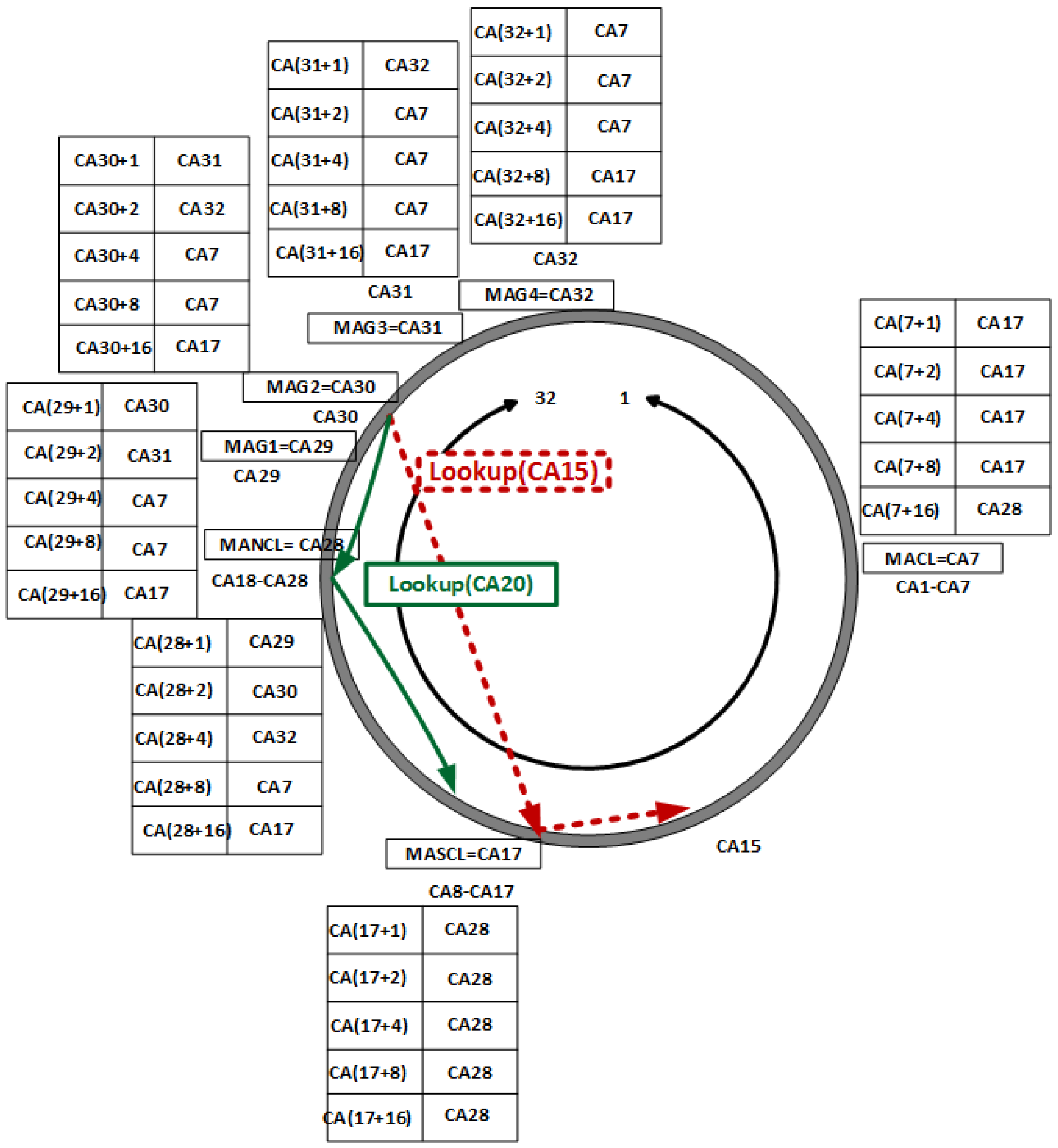
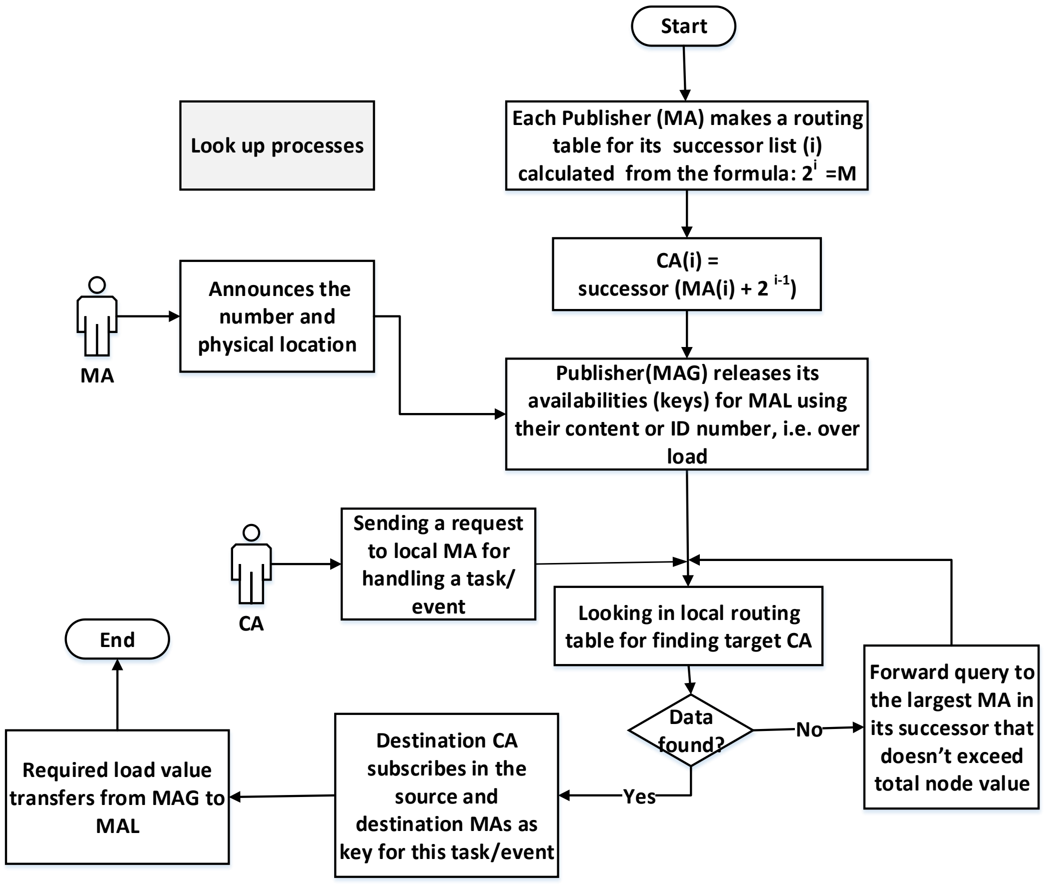


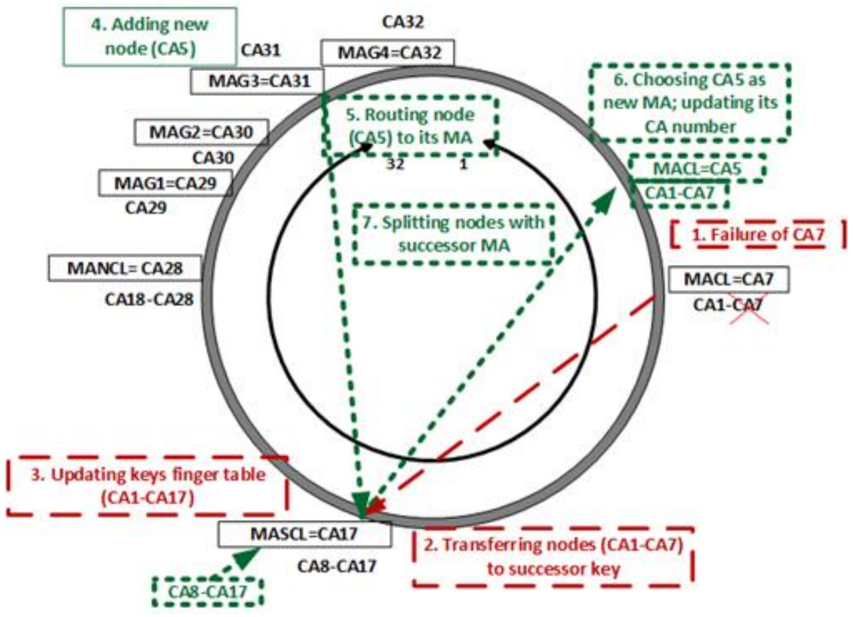
Figure 2.13. Lookup Process in DHT Algorithm (CA: Converter Agent, MA: Middle Agent)



Start
Look up processes
Each Publisher (MA) makes a routing table for its successor list (i) calculated from the formula: 2^i =M
CA(i) = successor (MA(i) + 2^{i-1})
MA
Announces the number and physical location
Publisher(MAG) releases its availabilities (keys) for MAL using their content or ID number, i.e. over load
CA
Sending a request to local MA for handling a task/ event
Looking in local routing table for finding target CA
Data found?
No
Forward query to the largest MA in its successor that doesn't exceed total node value
Yes
Destination CA subscribes in the source and destination MAs as key for this task/event
Required load value transfers from MAG to MAL
End

Figure 2.14. Lookup flowchart using DHT searching algorithm (CA: Converter Agent; MA: Middle Agent; MchAg: Matchmaker Agent; MAL: Middle Agent List)

One of the most important characteristics of DHT is its ability of self-healing and flexibility of scaling. It means the system automatically adapts to the arrival, departure, and failure of nodes. Thus a single point of failure will not necessarily affect the entire system. Furthermore, the system is robust in the case of adding nodes. Figure 2.15 displays the flexibility of an RCPC-32 system for handling the changes in the number of converters. In this Figure, the steps of handling a node failure and replacing it with another node is explained in detail. As mentioned before, the complexity of calculation is less than $(log_2 32)^2$ for handling peer joint/leave in DHTs.



4. Adding new node (CA5)
CA31
CA32
MAG4=CA32
MAG3=CA31
6. Choosing CA5 as new MA; updating its CA number
5. Routing node (CA5) to its MA
MAG2=CA30
CA30
MAG1=CA29
CA29
32
1
MACL=CA5
CA1-CA7
7. Splitting nodes with successor MA
1. Failure of CA7
MANCL= CA28
CA18-CA28
MACL=CA7
CA1-CA7
3. Updating keys finger table (CA1-CA17)
MASCL=CA17
CA8-CA17
2. Transferring nodes (CA1-CA7) to successor key
CA8-CA17

Figure 2.15. Management of Adding and Failure of Converter in a DHT model (Self-healing and Scalability)

2.6. Designing and Development of Publish-Subscribe over DHT for First Case Study

In this section, an MAS is introduced using publish-subscribe and a DHT for maintaining a sufficient voltage level during supply overload voltage. As depicted in Figure 2.1, each of the 32 nodes is connected to a converter agent through a converter, and a group of CAs based on system topology is assigned to each MA. Table 3.1 introduces seven MAs and their associated CAs in RCPC-32. MAs are categorized based on their node types including critical load (CL), semi-critical load (SCL), non-critical load (NCL), and distributed generator (DG). Publishers and subscribers exchange messages during their matching process. Figure 2.16 illustrates a message structure included the following segments: 1) a converter agent number (CAn) which is an integer between 1 and 32, 2) an Sid(Subscriber Id) for subscriber message types (3.4), or an Eid (Event Id) for publisher message types (3.5). At the last part of message, attributes and their values using operators have specified (3.4)-(3.11).

Table 2.1. Definition of Middle Agents for Case Study

Node ID	Key Nodes	Successor Nodes	Type of Nodes
1	MACL	CA1, …., CA7	Critical Load (CL)
2	MASCL	CA8, …., CA17	Semi-Critical Load (SCL)
3	MANCL	CA18, …., CA28	Non-Critical Load (NCL)
4	MAG1	CA29	DG1: Main Generator(MG)
5	MAG2	CA30	DG2:MG
6	MAG3	CA31	DG3:Auxilary Generator(AG)
7	MAG4	CA32	DG4:Capacitor Bank (CB)

Load balancing is applied to the system in order to cope with the active power contribution of the nodes after load changes. In this case study, the usage of agent technology for load balancing is described on the RCPC-32 system. Table 3.2 displays the numerical values and descriptions of 32 nodes in sample system.

CAn	**Sid_{ijk}/ Eid_{ijk}**	**Min Value ≤ ΔP ≤Max Value**

Figure 2.16. Publish-Subscribe Message Structure (CAn :Converter Agent Number, Sid: Subscriber/Consumer Id, Eid: Publisher/Event/Producer Id)

$$Eid_{ijk}\ (1 \le i \le 7, 1 \le j \le 4, 1 \le k \le 8) \tag{3.4}$$

$$Sid_{ijk}\ (1 \le i \le 7, 1 \le j \le 4, 1 \le k \le 8) \tag{3.5}$$

Where:

i is node type, i.e., CL, SCL, and NCL, DG1, DG2, DG3, DG4

j is physical area of nodes which is considered as four equal slides of microgrd based on its physical shape. e.g. A circle shaped microgrid can divide among four equal quarter
k is priority based on arrival time of message

Table 2.2. Nodes' Value and Description for Case Study (CAn: Converter Agent Number, CL: Critical Load, SCL: Semi-Critical Load, NCL: Non-Critical Load)

CAn	Node Type	P(kW)	CAn	Node Type	P(kW)
1	CL	20	**17**	SCL	25
2	CL	30	**18**	NCL	3
3	CL	110	**19**	NCL	11
4	SCL	2	**20**	NCL	38
5	SCL	5	**21**	NCL	24
6	SCL	3	**22**	NCL	3
7	SCL	135	**23**	NCL	5
8	SCL	40	**24**	NCL	2
9	SCL	1.2	**25**	NCL	83
10	SCL	10	**26**	NCL	7
11	SCL	40	**27**	NCL	49
12	SCL	95	**28**	NCL (PL)	12000
13	SCL	50	**29**	DG1(MG)	36000
14	SCL	20	**30**	DG2(MG)	41000
15	SCL	95	**31**	DG3(AG)	4000
16	SCL	1.2	**32**	DG4(CB)	520

The following steps are taken by agents for finding the match pairs of events and subscriber among all the 32 nodes. A publish-subscribe algorithm over a DHT infrastructure is applied for finding the match nodes:

1. Both of MG-related agents subscribe to the load MAs (MACL, MASCL, MANCL) (3.6), (3.7). As Figure 2.17 displays, six total message counts for this step are calculated.

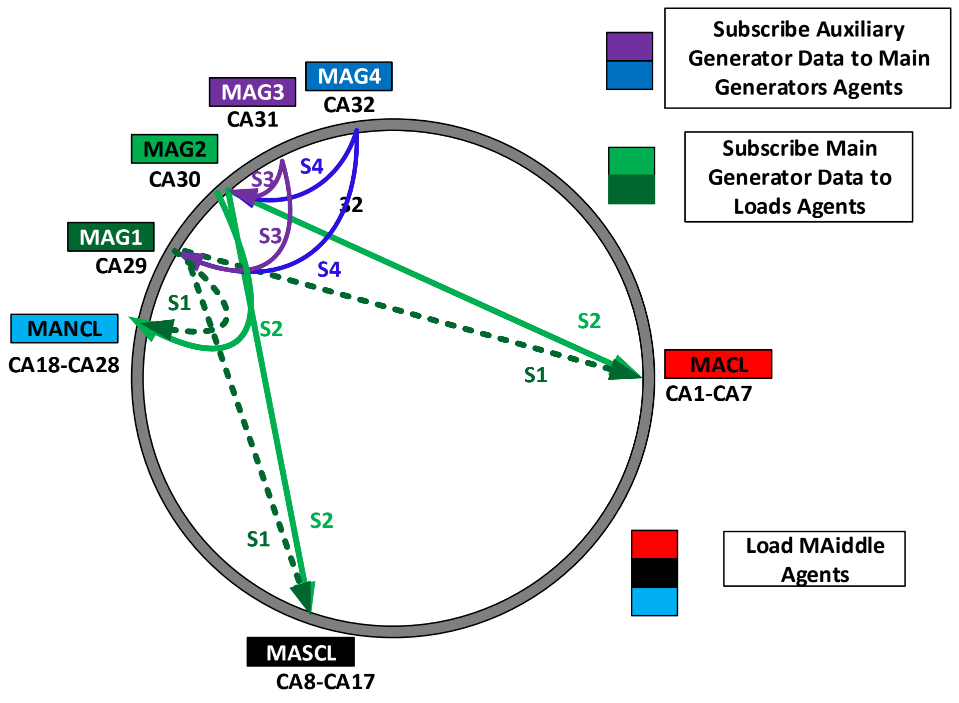
2. Auxiliary power resources subscribe to both main power resources (3.8), (3.9) and add four more message counts to the system complexity (Figure 2.17).

$$S1 = (Sid_{411},\ 0 \leq \Delta P \leq 200) \tag{3.6}$$

$$S2 = (Sid_{432},\ 0 \leq \Delta P \leq 100) \tag{3.7}$$

$$S3 = (Sid_{621}, 0 \leq \Delta P \leq 4000) \tag{3.8}$$

$$S4 = (Sid_{741}, 0 \leq \Delta P \leq 520) \tag{3.9}$$



Subscribe Auxiliary Generator Data to Main Generators Agents
Subscribe Main Generator Data to Loads Agents
Load MAiddle Agents
MAG4
CA32
MAG3
CA31
MAG2
CA30
MAG1
CA29
MANCL
CA18-CA28
MACL
CA1-CA7
MASCL
CA8-CA17
S1
S2
S3
S4

Figure 2.17. Matching Process between Publishers-subscribers (First step)

3. Load events realise from converter agents at an overload situation that happens whenever the load value increases more than 15%, which is defined as the

threshold for the test system, so the generator needs to provide extra power. Four different loads (# 3, 15, 20, and 23) experience the overload at the same time. Converter agents route the events (E1, E2, E3, and E4) to the related load MAs based on the first digit of their Eid numbers. So E1, E2, and E3, and E4 go to MACL, MASCL, and MANCL respectively (3.10)-(3.13). Routing happens based on the numerical coefficient of Sid and Eid message IDs (Figure 2.18)

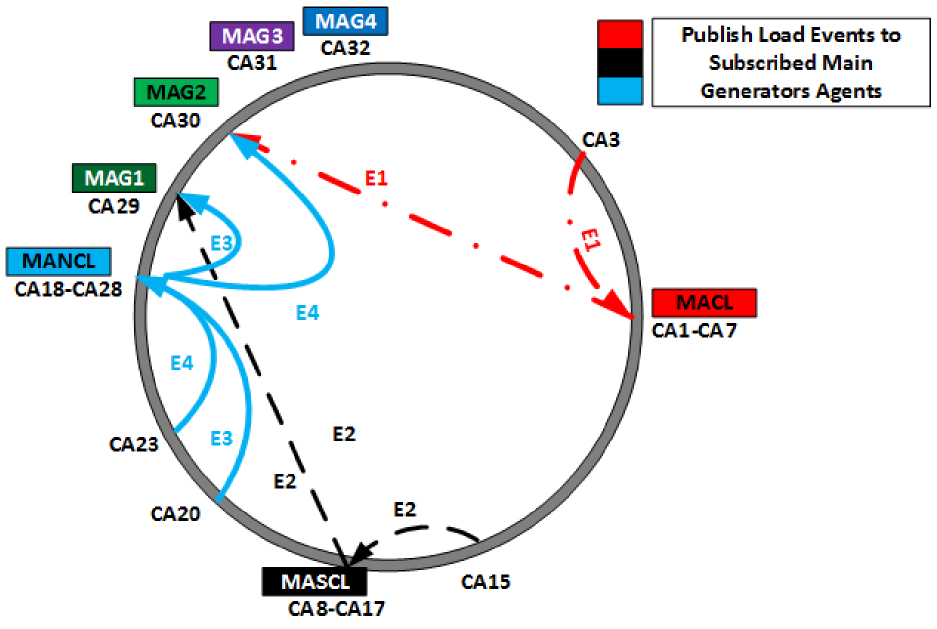


Figure 2.18. Matching Process between Publishers-subscribers (Second step)

$$E1 = (Eid_{111},\ \Delta P = 16.5) \quad (3.10)$$

$$E2 = \ (Eid_{221}, \Delta P = 14.25) \quad (3.11)$$

$$E3 = \ (Eid_{341}, \Delta P = \ 0.57) \quad (3.12)$$

$$E4 = \ (Eid_{321}, \Delta P = 0.75) \quad (3.13)$$

$$E5 = (Eid_{431}, \Delta P = 3600) \quad (3.14)$$

where ΔP is the absolute value of load changes:

$$\Delta P = |P_{current} - P_{nominal}| \qquad (3.15)$$

a) All of the load converter agents divide the Eid messages among three load MAs based on their first coefficient values that it can be 1 for critical loads (CL), 2 for semi-critical loads (SCL) and 3 for non-critical loads (NCL) from NodeID in Table 3.1.

b) Consequently, load MAs forward the messages to the physically closest generator by checking the second coefficient value.

c) In the subscriber MAs (MAG1 and MAG2), events are listed in a queue based on their load type and their arrival order, which can be specified using the first and third coefficient value of their Eid number.

d) The arrival coefficient (3rd number) sets in each agent individually; this means that for each publisher and subscriber agent there is a queue where events will be place based on their arrival time to that specific agent. In this step another 10 messages are added to the system message count.

4. After completing matching steps, each generator agent includes a routing table with publisher event IDs. As the infrastructure is made on DHT, the complexity of routing lookup messages according to the previous section is at most equal: $(\mathrm{O}(\log_2 n) = \mathrm{O}(\log_2 32) = 5$) message counts, which are very efficient compared to the other searching algorithms. Since four events are matched in this part, six message counts are added to the previous system complexity based on system metrics (Figure 2.19). CA3 and CA15 receives their required extra power from associated generators.
5. The generator agents issue an event if the generator power falls less than 10% of its normal (average) value. As Figure 2.20 illustrates, E5 is realized in this situation (3.14).
6. Based on the algorithm definition, both S3 and S4 subscribers are eligible to provide the required power for the system because they are both physical neighbors of MAG1 and are subscribed to its agent. Although considering the ΔP values of both subscribers, S3 has priority and it is matched to the E5. In this step, a total of two message counts are added to the system complexity. Therefore, in the whole process of the case study, a few tens of messages are delivered using agents for a system with tens of converters with no limitation in the number of events which might happen concurrently.

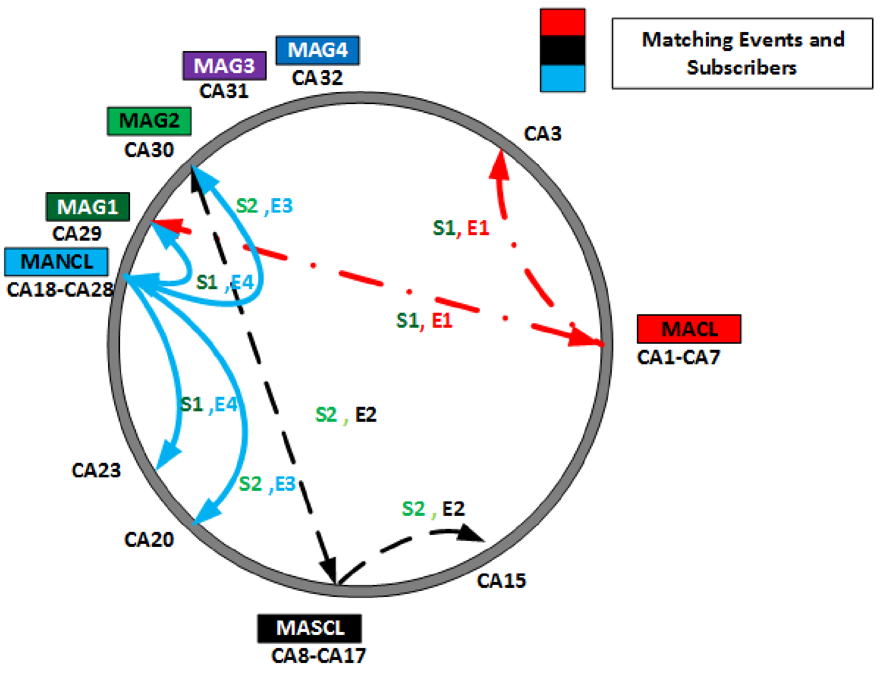


Figure 2.19. Matching Process between Publishers-subscribers (Third Step)

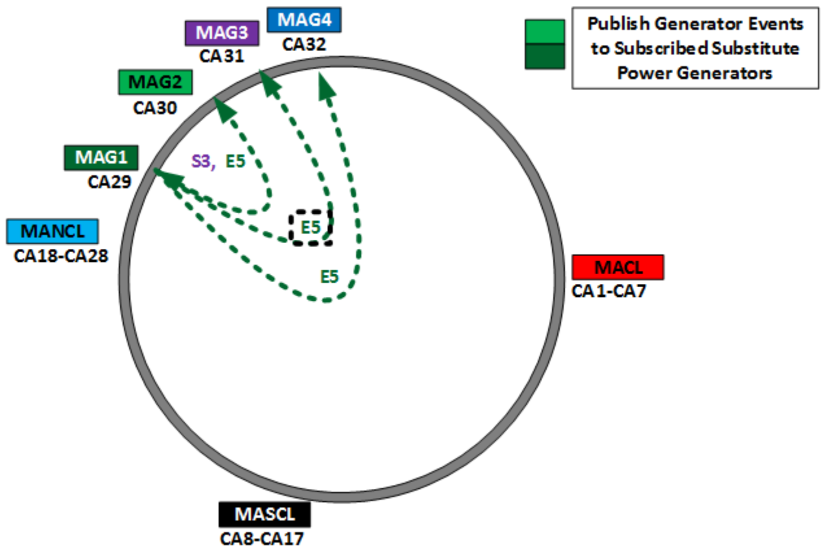


Figure 2.20. Matching Process between Publishers-subscribers (Fourth Step)

In this case study, the results of modeling a multi-agent system using different algorithms are analyzed numerically. It was found that the combination of publish-subscribe and DHT searching methods were the most efficient, scalable optimization algorithms for the defined medium-sized microgrid. As shown, an increase in the number of converters from 1 to 50 slightly affected the upper time limit of message exchange among converters. Furthermore, the management of adding and deleting converters based on a DHT infrastructure displays the same level of efficiency as the lookup process.

As a result, in section 4.2, a topic-based publish-subscribe method is modeled using the Jason Platform and the topic is chosen based on nodes' IP Addresses. However, since a topic-based model has been advanced to a content-based publish-subscribe model because of its advantages and flexibility, the Jason platform is not a suitable platform for implementing the case study. JADE is identified as the most flexible and capable agent platform which is also compatible with the FIPA standard [78]. Therefore it is used for developing a multi-agent control system in this thesis, and design processes and simulation results will be presented in the following chapters.

Chapter 3. Topology of DC Microgrid and Optimization Algorithm (Second Case Study)

Figure 3.1 displays the DC shipboard microgrid that is selected as a testbed for this research. This system is one of the most complicated cases to be addressed because it is islanded, has all sources connected via converters, and includes energy storage units. Islanded mode means that, in the case of scheduled or forced isolation, the microgrid must have the ability to operate stably and autonomously.

Islanded mode is challenging since a high inertia grid is not present. In addition, since all the sources are interfaced through power electronic converters, they can have very low inertia, resulting in fast control time constants compared to large synchronous machines. Finally, the presence of energy storage means that some sources can be considered bidirectional (charging and discharging). The system is basically a smaller size of MG-32 which is called MG-8 because it is including eight converters. The associated agent-based control system is called RCPC-8 that is designed based on publish-subscribe agent technology over DHT searching algorithm. System functionality and details are explained in section 3.1.

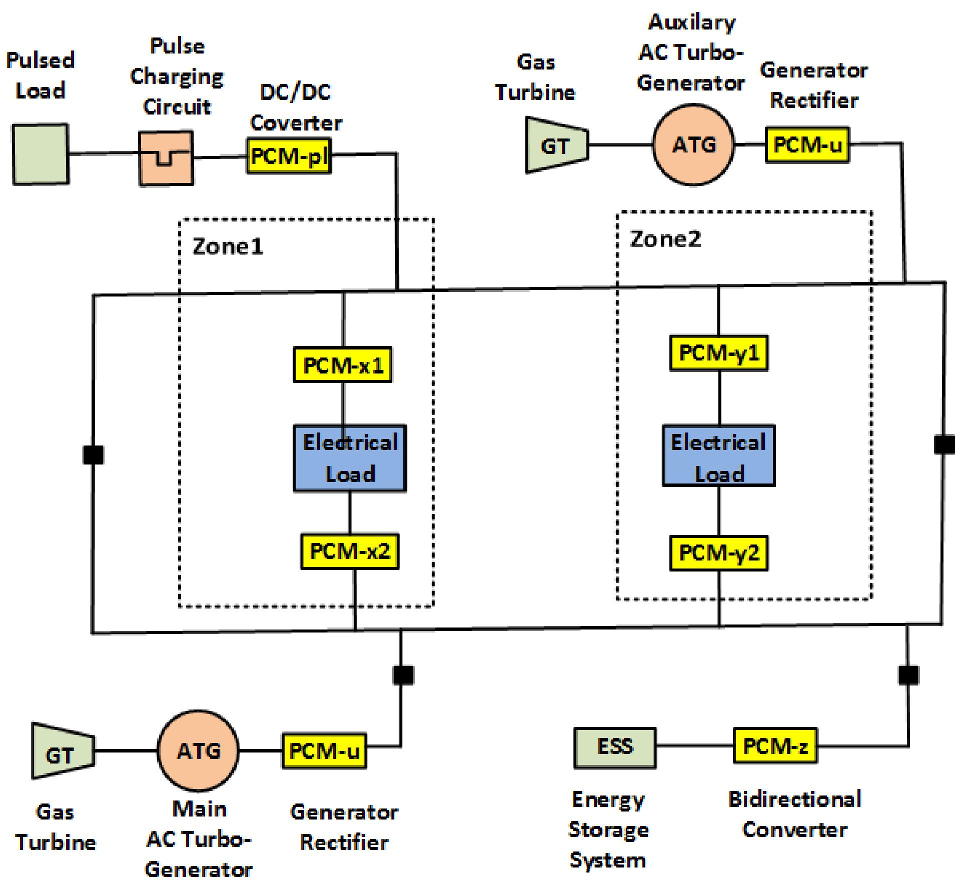


Figure 3.1. Architecture of a Simplified DC Shipboard Microgrid

3.1. System Configuration

In order to validate the distributed optimization and control method presented in this thesis, a simplified shipboard DC power system is used for case studies. The results of application of centralized control on this system are presented in [79], so it is an appropriate model to evaluate the agent-based control system developed in this thesis. The shipboard system is an isolated microgrid with converters between all sources of energy and the main buses as well as between all load centers and the main buses [80]. The example system, shown in Figure 3.1, has both fuel-based generators (of different ratings) and an electrochemical Energy Storage System (ESS). ESS can serve the microgrid both as source and load depending on the system need and battery State of Charge (SOC) condition [81]. The microgrid has two zones of utility loads and two converters referred to as Power Conversion Modules (PCMs) in each of the zones share the zonal load demand.

A bus is a conductor that connects multiple circuits or loads to a common voltage supply. Large loads and power distribution panels are connected to a bus via a breaker that is designed to trip the power flow in an overcurrent situation. There are different types of breakers in shipboard systems including bus-tie, load beakers that control the current of different buses such as Port side and Starboard side buses. Cross-tie breaker works when an imbalance load on one bus can affect the other bus, and possibly resulting in loss of all power.

Zonal and main bus level control systems enable flexible routing of energy within the test system. Each zone is managed by a zonal level control with a master-slave sharing scheme. The zonal PCM converter designated as the master regulates in-zone voltage while the slave PCM converter tracks a designated percentage of the master converter's output (sharing percentage). System control above the zonal level designates which converter is the master as well as the sharing percentage. Sharing of zonal load may vary from 0% to 100% depending on the system level optimizer decision. The PCMs connected with the load center are assumed to be unidirectional converters.

Control of the system energy flow above the zonal level is accomplished by the main bus level control. Within the main bus level control, a bus cluster control system regulates the total bus-tie current for the sum of all parallel bus-tie branches connecting two bus clusters. Thus the system level control can dictate how energy flows into each zone and how energy flows across the bus-tie. Two main buses (Starboard side bus and Port side bus) are connected by two cross-tie disconnects. These disconnects are used to connect the two bus-bars, control flow of inter-bus energy, maintain voltage levels, and disconnect them as necessary.

To demonstrate energy flow across the shipboard system, the ESS and Pulsed Load are located on different buses. Each zone or load center would introduce one variable (sharing variable; if one zonal converter carries (x)% of the zonal load, then the other converter would carry (1-x)% for system optimization. Storage as well as bus-tie current also introduces variables into the cost function. These variables determine the role of ESS, the inter-flow of energy between the two main buses, and the branch energy flow, and thereby determine the generators' operating points. An optimization algorithm will be presented that dynamically determines the global optimal values of these discretized variables in a distributed fashion in order to minimize system losses.

3.2. Requirements for Converter Coordination by Multi-Agent Control in a Microgrid

Hossain and Ginn presented a unique optimization algorithm for real-time distributed coordination of power electronic converters In DC microgrids [82]. Optimal power sharing ensures minimum transmission and distribution loss while enforcing constraints such as sources' ramp rates and capacity limits. The algorithm prunes off a significant number of search trees, distributes work load among the distributed control evenly, and determines the global optimal solution every time. An even distribution of work load makes the parallel system very reliable from a synchronism point of view. It offers scope to use other aggressive pruning off methods within its structure, which makes it useful for larger power systems as well. Aggressive truncations may not provide globally optimal solution always, but guarantee closer sub-optimal set-points. Convergence is ensured in any case. This algorithm is envisaged as part of a group project that aims at increasing the efficiency of the sample DC microgrid by controlling different parameters such as decreasing the convergence time among converters. In the following two sections, the usage of algorithms for microgrid system is described.

3.2.1. System Loss

In a power grid, system loss is mainly due to fuel usage inefficiency and the waste of energy due to the impedance of the transmission line. Determining fuel usage set-points is commonly called Economic Dispatch Problem (EDP). The objective of EDP is to schedule the committed generating unit's output so as to meet the required load demand at minimum cost, satisfying all unit and system operational constraints. The EDP problem can be generally expressed as:

$$P_{fuel} = aP^2 + bP + c$$

$$where\ a, b, and\ c\ are\ constants, and\ P\ is\ Electric\ power\ (kW), and$$

$$P_{fuel}\ is\ Power\ Loss\ caused\ by\ fuel\ consumption\ (kW)$$

$$P_{fuel} = a(VI)^2 + b(VI) + c$$

$$P_{fuel} = (aV^2)I^2 + (bV)I + c \qquad (2.1)$$

$where\ the\ only\ variable\ is\ the\ current\ flow\ if\ the\ generators'\ voltage\ magnitudes\ are\ close\ enough$ [83].

The waste of energy due to the impedance of the distribution system is called transmission loss. In a power system where there are many sources, loads, storages, and numerous transmission lines, transmission loss minimization becomes a strongly coupled optimization problem. Electrical conversion of the example microgrid of Figure 3.1, assuming idealized behavior of the lower level converter control systems, is shown in Figure 3.2. Transmission loss of the microgrid can be expressed as:

$$\begin{aligned}P_{tr} = {} & x^2I_1^2(R_1 + R_1' + R_2 + R_2') + y^2I_2^2\,(R_2 + R_2') \\ & + z^2I_3^2\,(R_2 + R_2' + R_3) + u^2I_4^2\,(R_2 + R_4) \\ & - 2xI_1(I_1R_1' + I_1R_2' + I_2R_2' + I_{PL}R_1' + I_{PL}R_2') \\ & - 2yI_2(I_1 + I_2 + I_{PL})R_2' - 2zI_3(I_1 + I_2 + I_{PL})R_2' \\ & + 2xyI_1I_2(R_2 + R_2') + 2xzI_1I_3(R_2 + R_2') - 2xuI_1I_4R_2 \\ & + 2yzI_2I_3(R_2 + R_2') - 2yuI_2I_4R_2 - 2zuI_3I_4R_2 \\ & + Constant \qquad (2.2)\end{aligned}$$

$$where\ P_{tr}\ stands\ for\ the\ transmission\ loss,$$

$$and\ R_i\ \ is\ \ the\ value\ of\ line\ resistance\ between\ each\ bus\ and\ converter$$

The constraints are:

$$\begin{cases} xI_1 + yI_2 + zI_3 + u\,I_4 \leq Main\_Generator\_Capacity \\ (1-x)I_1 + (1-y)I_2 + \ I_{PL} - zI_3 \leq Auxilary\_Generator\_Capacity \\ Main\_Gen\_Setting_t \leq Main\_Gen\_Setting_{t-1} + 0.1\,pu \\ Aux\_Gen\_Setting_t \leq Aux\_Gen\_Setting_{t-1} + 0.1\,pu \end{cases} \tag{2.3}$$

Here I_1 is the load of Zone1, I_2 of Zone2, I_{PL} of pulsed load. I_4 is the maximum charging/discharging current of ESS and I_3 is the maximum allowed cross-tie (inter-bus) current. The variables x, y, z, and u are ratios that would determine the flow of energy through all the branches and must be optimized by the system controllers in a distributed fashion that would ensure minimal loss of the system. The value of x and y may vary from 0 to 1 with a step size of 0.1 (as zonal converters are unidirectional). On the other hand, z and u may have any value between -1 and +1 with a step size of 0.2 (as bidirectional).

The EDP problem as seen in (2.1) can be readily consumed into the transmission loss problem of (2.2) without any approximation. So, if the problem in (2.2) can be globally optimized, transmission loss plus EDP also can be optimized altogether. In section 3.2.2., the optimization of transmission loss will be described in detail.

3.2.2. Optimization Algorithm

As was mentioned in section 1.3, the quality of distributed control technologies are compared based on some parameters such as scalability, fault management, and convergence rate. Similarly, there are several properties such as required convergence time and distribution rate that measure the quality of a distributed optimization algorithm. In any real-time dynamic system, the required time for convergence is one of the most important properties to be considered. Failure to converge within the time boundaries means that the system would run with old settings for an indeterminate interval which may violate constraints and destabilize the grid. If the algorithm doesn't ensure convergence to optimal points, then its efficiency becomes low.

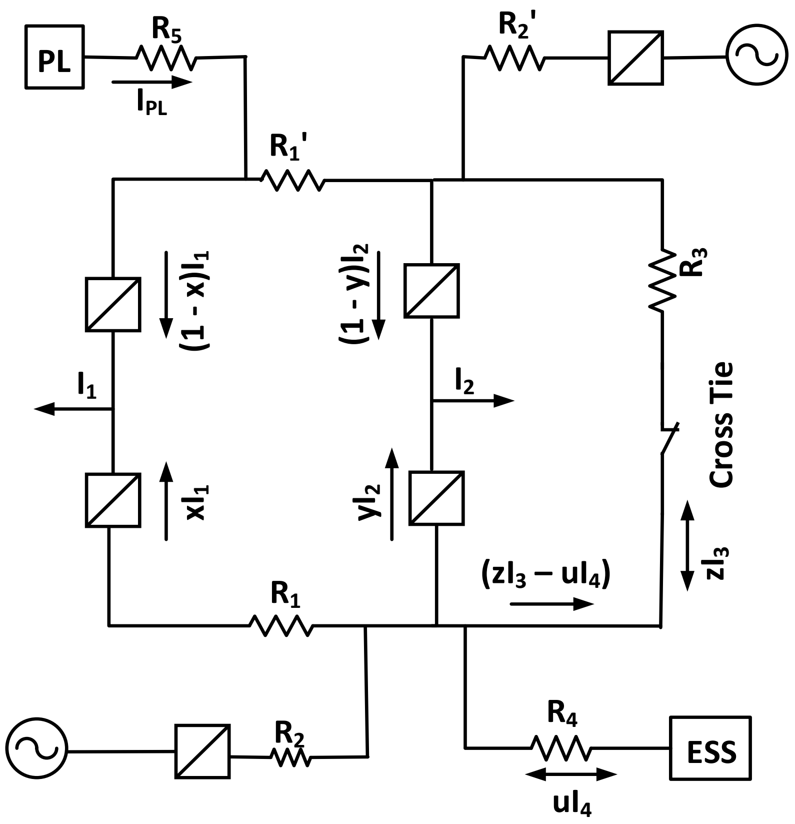


Figure 3.2. Electrical Representation of the Sample DC Microgrid

Finally, the distribution rate of workload ensures system expandability within the time limit. Keeping all these in mind, a unique algorithm has been devised here that evenly distributes the optimization task among the intelligent software modules of EMS, while reducing the computational requirements significantly to ensure minimum time requirements. Optimality and convergence are guaranteed by this algorithm. If the microgrid has many zonal load centers and a large number of variables, some other pruning techniques such as “check and eliminate,” “reduction of variables,” and “sliding” can be adopted to ensure convergence with sub-optimality.

3.2.3. Scaling and Change of Variables

Hossain and Ginn calculated loss by using a quadratic function in which all the variables are coupled (2.4). As the variables are not loosely coupled, the basic objective function in the straightforward expression is not applicable for distributed control. This type of problem falls within the category commonly named Mixed Integer Problem (MIP). Depending on the nature of constraints, MIPs are sub-divided. MIP models with quadratic constraints are called Mixed Integer Quadratically Constrained Programming (MIQCP) problems. Models without any quadratic features are often referred to as Mixed Integer Linear Programming (MILP) problems. MIP models with a quadratic objective function but without quadratic constraints are called MIQP problems. As seen in equations (2.4), (2.5), (2.6) and (2.7), the constraints are not quadratic. Thus the above problem falls within the category of an MIQP problem. All the variables are tightly coupled with one another in (2.4). To alleviate this, a conversion process called "Scaling and Change of Variables" is used to make the system unidirectional interacting, which orients the variables as a leader-follower system. This method is briefly described below. The following cost function can be considered

$$\begin{aligned} P_{tr} &= Const + a_{11}x_1^2 + a_{22}x_2^2 + a_{33}x_3^2 + a_{44}x_4^2 + a_1x_1 \\ &\quad + a_2x_2 + a_3x_3 + + a_4x_4 + a_{12}x_1x_2 + a_{13}x_1x_3 \\ &\quad + a_{14}x_1x_4 + a_{23}x_2x_3 + a_{24}x_2x_4 + a_{34}x_3x_4 \end{aligned} \tag{2.4}$$

where x1, x2, x3, x4 are variables and the rest are load dependent constants. Let

$$z_1' = b_1x_1 \tag{2.5}$$

$$z_2' = c_1x_1 + c_2x_2 \tag{2.6}$$

$$z_3' = d_1x_1 + d_2x_2 + d_3x_3 \tag{2.7}$$

$$z_4' = e_1x_1 + e_2x_2 + e_3x_3 + e_4x_4 \tag{2.8}$$

Now using this 'z'_i' domain variables, the basic cost function can be converted as follows

$$P_{tr} = Const + z_1'^2 + k_1z_1' + z_2'^2 + k_2z_2' + z_3'^2 + k_3z_3' + x_4^2 + k_4z_4' \tag{2.9}$$

where

$$a_{11} = b_1^2 + c_1^2 + d_1^2 + e_1^{2`} \tag{2.10}$$

$$a_{22} = c_2^2 + d_2^2 + e_2^2 \tag{2.11}$$

$$a_{33} = d_3^2 + e_3^2 \tag{2.12}$$

$$a_{11} = e_4^2 \tag{2.13}$$

Again,

$$z_1'^2 + k_1 z_1' = \left[z_1' + \frac{k_1}{2}\right]^2 - \left[\frac{k_1}{2}\right]^2 = z_1^2 - \left[\frac{k_1}{2}\right]^2 \quad (2.14)$$

Where,

$$z_1 = [z_1' + \frac{k_1}{2}] \quad (2.15)$$

$$z_2 = [z_2' + \frac{k_2}{2}] \quad (2.16)$$

$$z_3 = [z_3' + \frac{k_3}{2}] \quad (2.17)$$

$$z_4 = [z_4' + \frac{k_4}{2}] \quad (2.18)$$

Equation (13) then converts to the form

$$P_{tr} = z_1^2 + z_2^2 + z_3^2 + z_4^2 + Const \quad (2.19)$$

Equation (2.19) is the converted cost function that needs to be optimized instead of (2.4) or (2.9). In this equation, z_1 is independent and z_2, z_3 & z_4 follow it with unidirectional dependency. All variables are squared in the converted cost function, which creates loss in the form of a parabola. It gives us an advantage to prune off a significant number of search trees. The loss, due to the variable z_1, takes the form shown in Figure. 3.3.

Loss due to z_1 is z_1^2 would always have a positive value with a parabolic shape as shown in Figure 3.3. The value of z_1, which makes the loss component minimum, is very important because all values of z_1 either on the right or the left to it would be pruned out. Which side values (either right or left to absolute minimum of z_1) would be pruned off, depends on the constraint. If the constraints are less than or equal, the right side values would be pruned off. If constraints are greater than or an equal type, it is the left side values. Pruning, as shown in Figure 3.3, reduces the iteration number significantly and does not introduce any approximation or sub-optimality. Each value of existing z_1 would cause to initiate a search tree, so another span of z_2 values would get a corresponding loss component of every z_1. Values of z_2 would also be pruned, as was done for z_1 values, and the same would be done for others.

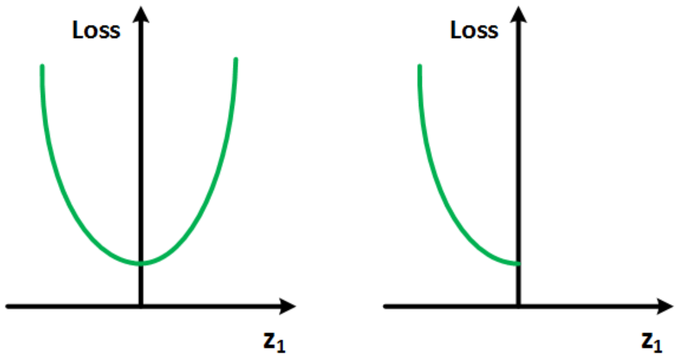


Figure 3.3. Loss Values in Two Situations of Normal Loss, applying an MIQP Sub-optimal Algorithm

Capacity constraint would truncate the span of z_4 values, and there would be no need to consider all the available values of z_4. Subsequently, the value of z_4 would be picked corresponding to the smallest value of z_4^2. One search tree is shown in Figure 3.4 along with pruning and an optimal solution. Red ones are pruned without introducing any approximation. It provides significant advantage in computation level. If there are four variables each with 11 steps as in the discussed case:

- Number of computation tree without pruning =11*11*11*11=14641
- Number of computation (around, as observed) tree with pruning <= 4*6*5*1=120

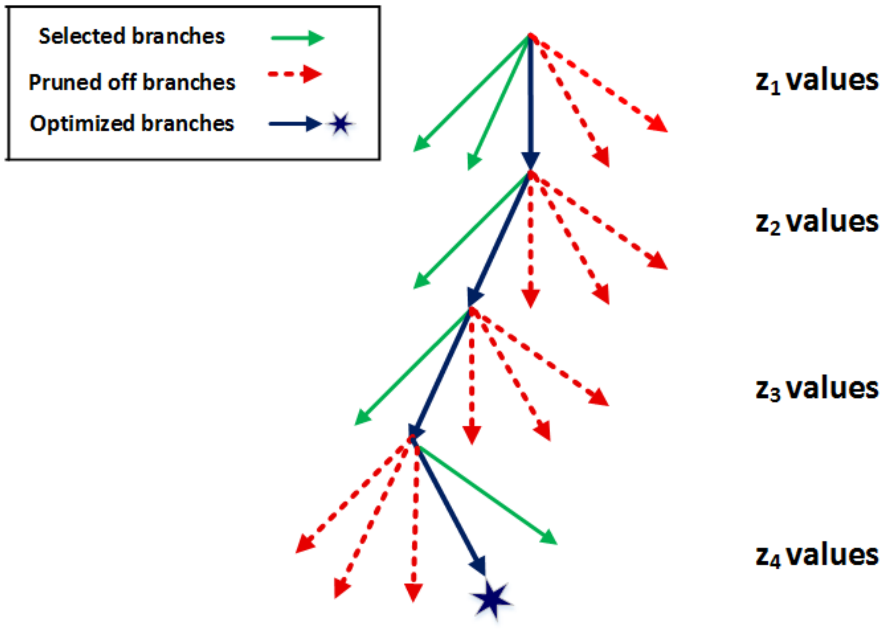


Figure 3.4. Running Optimization Formula Using Agents

There are some aggressive pruning techniques such as “Check and eliminate,” “Sliding over variables,” which can further truncate a search tree and provide either a global or closest sub-optimal solution. These aggressive centralized optimization algorithms will decrease the loss while running on top of an MIQP-distributed and sub-optimal solution. However, they apply centralized optimization on top of distributed optimization, which are not considered in this work.

Chapter 4. Designing a Publish-Subscribe Distributed Control Model over a DHT Searching Algorithm for Second Case Study

Based on the results extracted from chapter 2 on design and development the publish-subscribe agent platform (RCPC-32), the agent-based control platform for the second test study(RCPC-8) is designed and implemented. In the following two sections the procedures of choosing the most suitable software application for developing agent platform, and the design of UML classes using the selected software are described.

4.1. Exploring a Powerful Software Platform for developing Multi-agent Systems

Developing multi-agent system control model for power systems is a multidisciplinary research area involving three major disciplines of Power Systems, Telecommunication, and Computer Engineering. In each of these areas, the following sections are developed:

- Telecommunication Engineering
    - Assigning a suitable networking protocol for communication among agents such as TCP/IP;
- Computer Engineering
    - Designing System Use Cases;
    - Using a multi-agent development platform like JADE for developing agent-based control systems;
    - Designing an applicant based on system requirements for integrating power systems and agent-based control platforms, e.g., JMAtlink;
- Power Systems
    - Developing a power system model using a strong computational engine such as MATLAB.

Different technologies are used in academic and industrial systems to develop agent technologies. Some of the academic tools are: RT-lab [84], Matlab, and JADE, while three samples of industrial tools are: SICAM PAS, which is a power automation system developed in Siemens [85], [86], and Real Time Digital Simulator (RTDS) that is the world's benchmark for performing real time power system simulation [87]. An industrial sample called the KCP&L (Kansas City Power and Light Company) project is located in a number of Kansas City neighborhoods [88]. It is a five-year project established in 2009 that is funded by the U.S. Department of Energy ($48 million). The project benefits approximately 14,000 utilities customers in the area, with six main industrial vendors including Siemens. The main achievements of the project include: VAR control, voltage regulation, data management, outage management systems, and power quality to meet the range of customers who need energy efficiency.

Some companies market API platforms using publish-subscribe for developing agent technology including [89]:

- SIENA: Using Java and C# for topic-based;
- Hermes: Using XML and Java for topic and content-based;
- Xmessages: Using Java, C++ for topichannel- and content-based;
- Gryphon: Using Java Message Service (JMS) for topic and content-based.

For developing an RCPC-32 system, a group of open-source agent platforms are reviewed and evaluated for their compatibility with an RCPC-32 system design. In the following section, four of them are explained: JMS, Jason, NetLogo, and JADE.

4.1.1. Java Message Service over JBoss Server

The Java Message Service (JMS) agent gateway is a utility designed to allow agents within a JADE platform to interact with a JMS provider [90]. A JMS toolbox is one of the most common APIs for developing both topic-based and content-based publish-subscribe technologies. It can also be considered as a Point-to-Point messaging mode. JMS uses FIPA as an interaction protocol and JADE as an implementation protocol. The following steps have been completed for testing JMS in the RCPC-32 system.

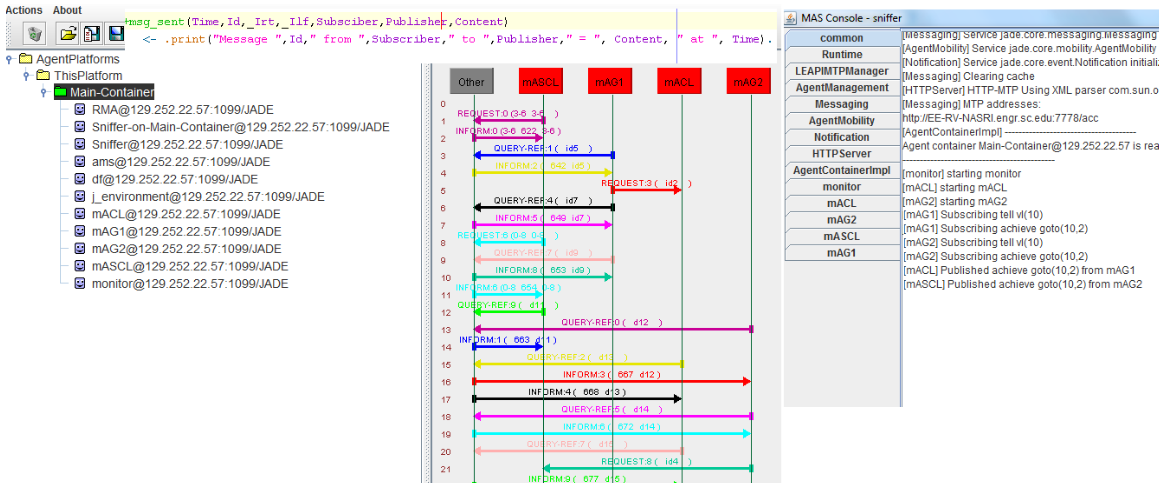
- JBoss v5.0 at localhost (192.127.0.1) which is installed and started

- Developing producer (publish) and two sample consumers (subscribe) using JMS:
    - ------Entering JMS TopicProducer-------
    - --------Entering JMS Example TopicConsumer1----
    - --------Entering JMS Example TopicConsumer2----
    - Incoming message: message 1 from TopicProducer-----
    - Incoming message: message 2 from TopicProducer-----
    - Incoming message: message 3 from TopicProducer-----
    - --------Exiting JMS Example TopicConsumer2----
    - --------Exiting JMS Example TopicConsumer1----

There are some pros and cons in using JMS as an agent developer for an RCPC-32 design. Facilitating implementation of publish-subscribe is one of the main advantages of JMS, but its non-flexibility for communicating with Simulink removes it from the candidate's list. The programming language is JADE although it requires a JBoss server for communication among publisher and subscribers, so it does not really meet the distributed control requirement for RCPC-32.

4.1.2. Jason

Another API that uses the JADE platform is Jason [91]. Jason is an interpreter for an extended version of AgentSpeak. It implements the operational semantics of that language and is available as open source. Jason provides a platform for the development of multi-agent systems with many customizable features for users. The case study discussed in section 3.1.3 for evaluation of a BDI model, was developed using the last version of Jason (v1.4.2) platform [92]. Three agents are communicating through a Jason platform using a bidding algorithm. Figure 4.2 displays a snapshot of communication among four agents for running a bidding algorithm in the sample MAS developed in a Jason platform. A bidding algorithm is cost effective for microgrids with less than 10 nodes, but it is not cost effective for larger microgrids, as is shown using numerical analysis results in Figure 2.9.



```
+msg_sent(Time,Id,_Irt,_Ilf,Subsciber,Publisher,Content)
   <- .print("Message ",Id," from ",Subscriber," to ",Publisher," = ", Content, " at ", Time).
```

```
RMA@129.252.22.57:1099/JADE
Sniffer-on-Main-Container@129.252.22.57:1099/JADE
Sniffer@129.252.22.57:1099/JADE
ams@129.252.22.57:1099/JADE
df@129.252.22.57:1099/JADE
j_environment@129.252.22.57:1099/JADE
mACL@129.252.22.57:1099/JADE
mAG1@129.252.22.57:1099/JADE
mAG2@129.252.22.57:1099/JADE
mASCL@129.252.22.57:1099/JADE
monitor@129.252.22.57:1099/JADE
```

```
0
1   REQUEST:0 (3-6  3-6   )
2   INFORM:0 (3-6  622  3-6 )
3   QUERY-REF:1 (  id5   )
4   INFORM:2 (  642  id5 )
5   REQUEST:3 (  id2   )
6   QUERY-REF:4 (  id7   )
7   INFORM:5 (  649  id7 )
8   REQUEST:6 (0-8  0-8   )
9   QUERY-REF:7 (  id9   )
10  INFORM:8 (  653  id9 )
11  INFORM:6 (0-8  654  0-8 )
12  QUERY-REF:9 (  d11   )
13  QUERY-REF:0 (  d12   )
14  INFORM:1 (  663  d11 )
15  QUERY-REF:2 (  d13   )
16  INFORM:3 (  667  d12 )
17  INFORM:4 (  668  d13 )
18  QUERY-REF:5 (  d14   )
19  INFORM:6 (  672  d14 )
20  QUERY-REF:7 (  d15   )
21  REQUEST:8 (  id4   )
    INFORM:9 (  677  d15 )
```

```
[Messaging] Service jade.core.messaging.Messaging
[AgentMobility] Service jade.core.mobility.AgentMobility
[Notification] Service jade.core.event.Notification initiali
[Messaging] Clearing cache
[HTTPServer] HTTP-MTP Using XML parser com.sun.o
[Messaging] MTP addresses:
http://EE-RV-NASRI.engr.sc.edu:7778/acc
[AgentContainerImpl] --------------------------------------
Agent container Main-Container@129.252.22.57 is rea
--------------------------------------

[monitor] starting monitor
[mACL] starting mACL
[mAG2] starting mAG2
[mAG1] Subscribing tell vl(10)
[mAG1] Subscribing achieve goto(10,2)
[mAG2] Subscribing tell vl(10)
[mAG2] Subscribing achieve goto(10,2)
[mACL] Published achieve goto(10,2) from mAG1
[mASCL] Published achieve goto(10,2) from mAG2
```

Figure 4.1. A Snapshot of Communication among Three Agents in a Sample MAS a Using a Jason Platform

Jason is a topic-based agent programming language that has been developed based on a JADE library. It is compatible with BDI agent architecture and it also is integrated with agent console and sniffer. The disadvantages of Jason are that it cannot be connected to MATLAB, which is used for developing the RCPC-32 microgrid model, and it is not compatible with publish-subscribe technology.

4.1.3. NetLogo

NetLogo is a multi-agent programmable modeling environment. Its programming language is a Logo dialect extended to support agents. It is used by tens of thousands of students, teachers, and researchers worldwide [93], [94], [95]. Since a NetLogo platform works individually and cannot integrate to the hardware devices or hardware simulators for developing power devices, it is required to define a function for each of them and control the functions using agents. As an advantage, it is compatible with publish-subscribe modeling because it has been around since 1999. But based on the concept of publisher, NeLogo doesn't accept a message structure for publisher and subscriber as defined in section 3.2.2. In addition, in this platform, it might not be possible to measure message count as metric value because it was not designed for real-time systems, so the quality of messages are more important than their numbers. Figure 4.3. displays a snapshot of running an Agentset Efficiency program in NetLogo. The application has optimized the arrangement of blocks using agent technology.

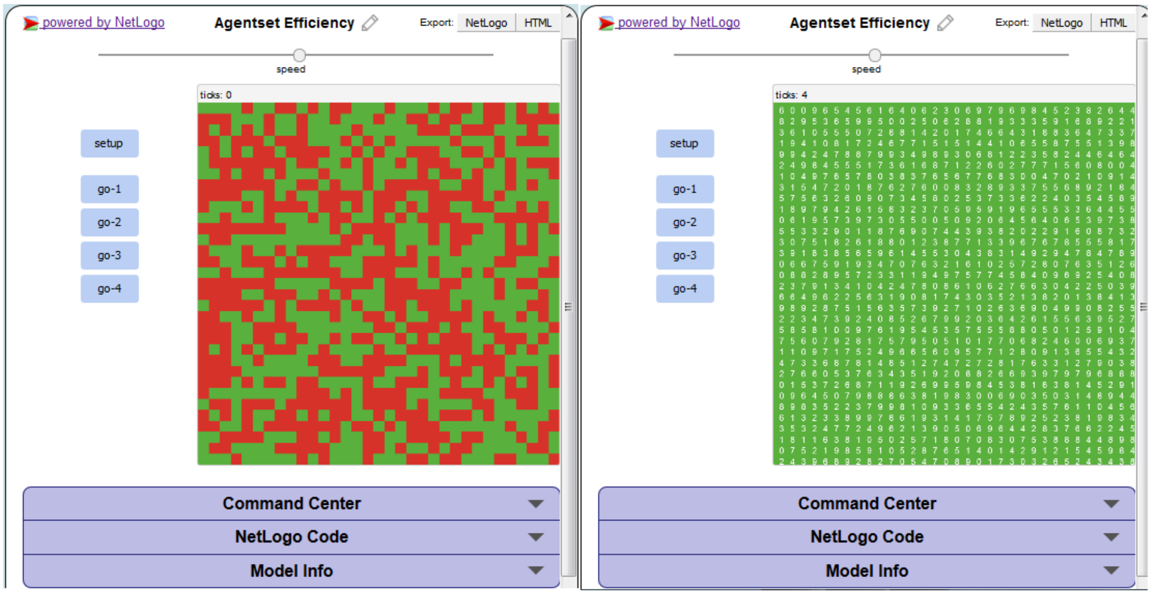


Figure 4.2. A Snapshot of Running an Agentset Efficiency Program in NetLogo

4.1.4. JADE

The Java Agent DEvelopment Framework is an open source platform for peer-to-peer agent-based applications. JADE is a software framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middleware that complies with FIPA specifications and through a set of graphical tools that support the debugging and deployment phases [96]. A JADE-based system can be distributed across machines (which do not even need to share the same operating system), and the configuration can be controlled via a remote GUI. The configuration can even be changed at run-time by moving agents from one machine to another, as required.

JADE is completely implemented in Java language and the minimal system requirement is the version 5 of Java (the runtime environment or the JDK). Besides the agent abstraction, JADE provides a simple yet powerful task execution and composition model, peer to peer agent communication based on the asynchronous message passing paradigm, a yellow pages service supporting a publish-subscribe discovery mechanism, and many other advanced features that facilitate the development of a distributed system 97.

Since JADE is capable of developing communication with hardware devices and simulators such as MATLAB/Simulink, it is a good platform for the RCPC system. In addition, JADE is compatible with a FIPA agent design and publish-subscribe agent technology. All of the Java libraries are accessible through JADE and increase the power of it. Although developing MAS using JADE is complicated because of its advantages, it has been selected as the agent platform for development of the RCPC system. Table 4.1 summarizes and compares the four aforementioned agent platforms. The evaluation is based on RCPC system requirements and the JADE platform is chosen as the most compatible one with the RCPC system design. RCPC-8, which is simplified version of RCPC-32, is selected as the test system for developing integrated platform.

Table 4.1. Comparison among some of the most well-known agent development platforms

S/W Platform for Developing MAS	Advantages	Disadvantages
JMS	Compatible with Publish-Subscribe Developed based on JADE library	Needs JBoss server to run Low flexibility for connecting to MATLAB
Jason	Developed based on JADE Library Integrated with Agent Console, and Sniffer Developed based on BDI model	Low flexibility for connecting to MATLAB and developing publish-subscribe Doesn’t accept a message format for publish-subscribe

S/W Platform for Developing MAS	Advantages	Disadvantages
NetLogo	Compatible with publish-subscribe, but its concept for publisher means multicast Integrated with Agent Console	A function for each of the devices should be defined to replace Matlab model
JADE	Flexible for adding any toolbox to connect to H/W Devices/Simulators Compatible with publish-subscribe Developed based on FIPA standard Integrated with Agent Console, and Sniffer	Hard to develop because all the communication classes and console should be developed

4.2. Designing JADE Classes based on Publish-Subscribe design for Second Case Study

As explained in section 3.4, the subscribers design pattern is made of {attribute, operator, value} tuples, where operators can be one of {<, =, >, ≤, ≥}, and the publisher is a set of {attribute, value}. Since Ag1, the agent that associates to the first search tree based on design, usually generates the minimum confident values, it is chosen as subscriber. The other agents calculate z_i values for their search trees and publish to subscriber agents only if they meet the conditions in (5.6), (5.7), and (5.8). Each search tree is denoted by $ST_{\ i}$ where $i \in S = \{a \mid a\ is\ one\ of\ the\ z_{\ i}\ values\}$

$$x = f(z_1) \tag{5.1}$$

$$y = f(z_1, z_2) \tag{5.2}$$

$$z = f(z_1, z_2, z_3) \tag{5.3}$$

$$u = f(z_1, z_2, z_3, z_4) \tag{5.4}$$

Definition of subscriber agents:

$$Ag1 = (ST_1 \mid x \le z_1,\ y \le z_2,\ z \le z_3,\ u \le z_4) \tag{5.5}$$

Definition of publisher (event) agents:

$$Ag2 = (ST_2|\ z_1,\ z_2,\ z_3,\ z_4) \tag{5.6}$$

$$Ag3 = (ST_3|\ z_1,\ z_2,\ z_3,\ z_4) \tag{5.7}$$

$$Ag4 = (ST_4|\ z_1,\ z_2,\ z_3,\ z_4) \tag{5.8}$$

Graphical model of publish-subscribe agents applying (5.1)- (5.8) is shown in Figure 4.4. The searching area of subscribe agent (Ag1) is decreased from big rectangular in each period of time to the smaller rectangular of S1, and S2. As a result the publisher Agents (Ag2, Ag3, And Ag4) are able to publish their service request using P1 and P2 to the limited areas of S1, and S2 respectively.

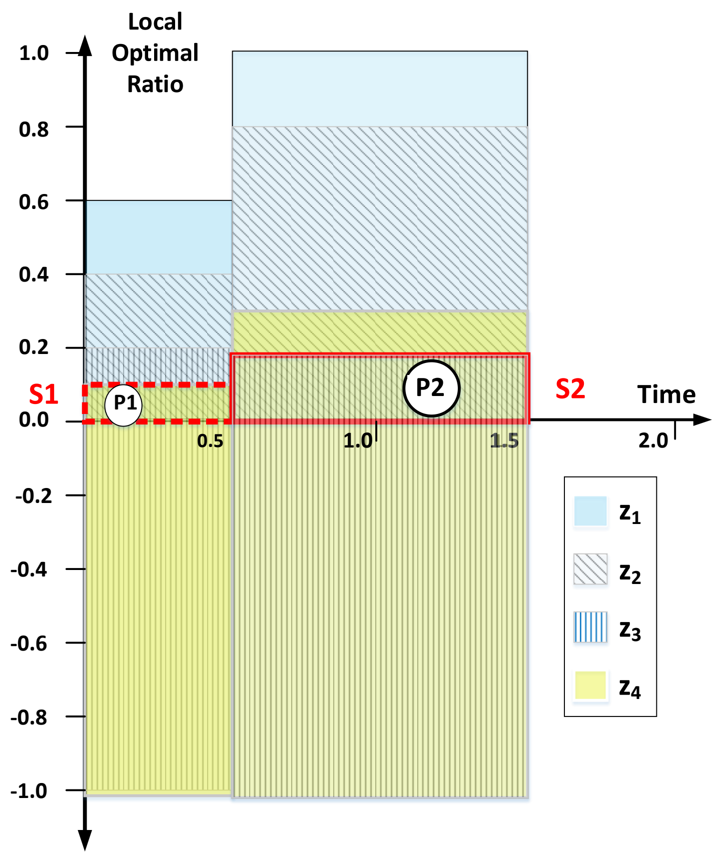


Figure 4.3. Application of Publish-Subscribe for Finding Global Optimization Ratios

In this JADE platform, based on publish-subscribe design a couple of main classes are developed for clarification of agent behaviors. For each agent, one individual Java class is created to communicate with other agent classes and exchange data based on the publish-subscribe algorithm. Figure 4.4 illustrates two agent classes and three main Java classes (AgentCoordinator, filterBehaviour, and monitorBehaviour) including their attributes and methods. Agent classes are represented with Agent1 and Agent2 in this case study. Each agent class gets its associated converter load through a signal attribute. As shown, *signal1* and *signal2* attributes are associated to Agent1 and Agent2 respectively. The message contents are transferred through these signals using filter behavior. In addition, variables "agentsSubscribed" and "agentsPublishing" work based on monitor behavior to count the number of message exchanges.

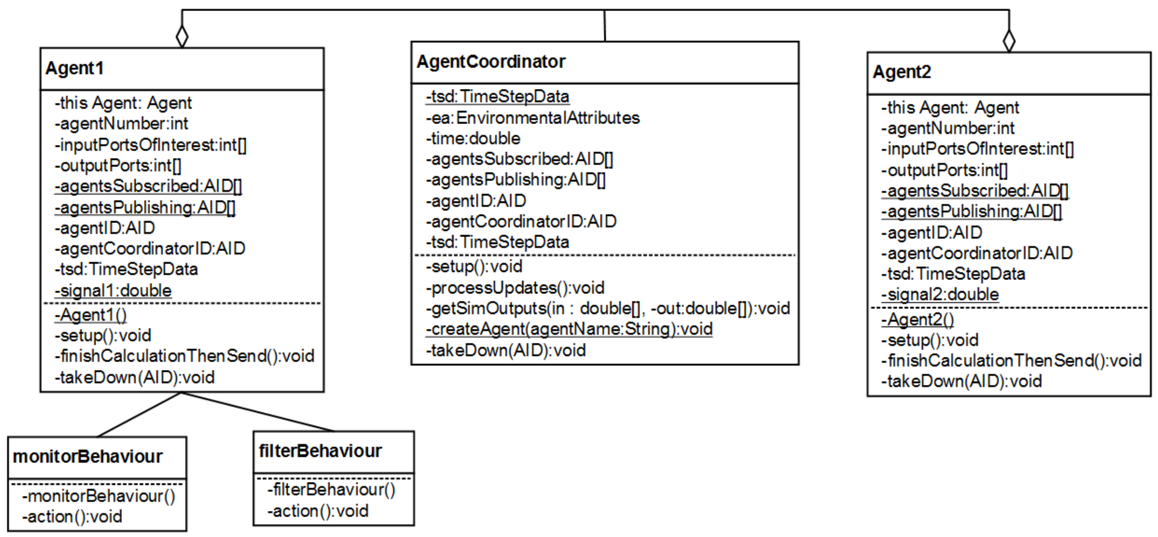


Figure 4.4. UML of Communication between Java Classes in a JADE Platform

Different Java classes are used for developing the RCPC-8 controller platform including a couple of core classes for the coding agent environment. These core classes are: AgentCoordinator, AgentServer, TimeStepData, and UsefulAgentMethods. Also there are specific agents' classes defined based on each agent's task named as Agent1, Agent2, Agent3, and Agent4. These classes are illustrated in the Unified Modelling Language (UML) diagrams of Figure 4.5. UML is a standard used for representing the code structure of computer programs. AgentCoordinator is one the main components of the agent environment, which acts as a bridge between Simulink and the ATF. It is instantiated as an agent itself for ease of communication in the JADE framework, and so there is the possibility to sleep while idle.

The AgentCoordinator, in its setup method, registers its service with the DF of JADE, which is the provision of updates to agents when new data arrives. It also starts up the AgentServer and then initializes the ATF. Through its monitorBehaviour, and with the help of the DF, the AgentCoordinator maintains communication links with the agents wishing to receive incoming data. When the AgentCoordinator receives an array of data from Simulink, it broadcasts this to the agents registered to its service. When these agents have finished working with the data, they return it back to the AgentCoordinator, which passes the processed data on to Simulink.

The other core class, which is critical for the RCPC-8 real-time system, is TimeStepData class. One restriction currently imposed by the model is that the agents have to work in synchrony with each sample period of Simulink. This was done for ease of implementation rather than out of necessity. Given that the agents finish their operations over various lengths of time, the AgentCoordinator instantiates TimeStepData as an object to store information as and when agents from the Agent Task Force (ATF) send their processed data back. The AgentCoordinator stores new information, arriving from Simulink in this object, and then sends a copy of it out to all the agents. On receiving the TimeStepData object, the agents process the information it contains, update its contents, and return it.

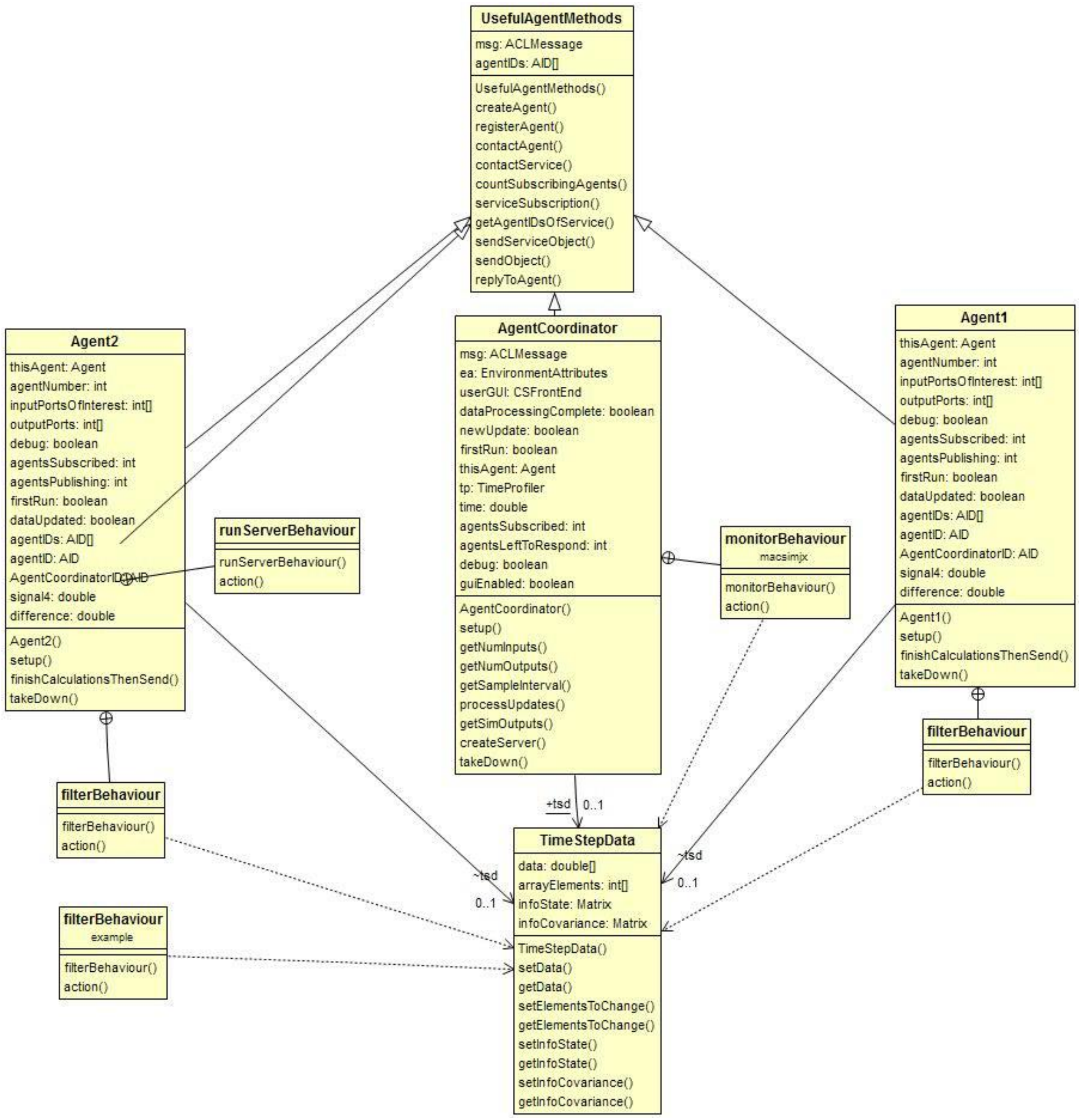
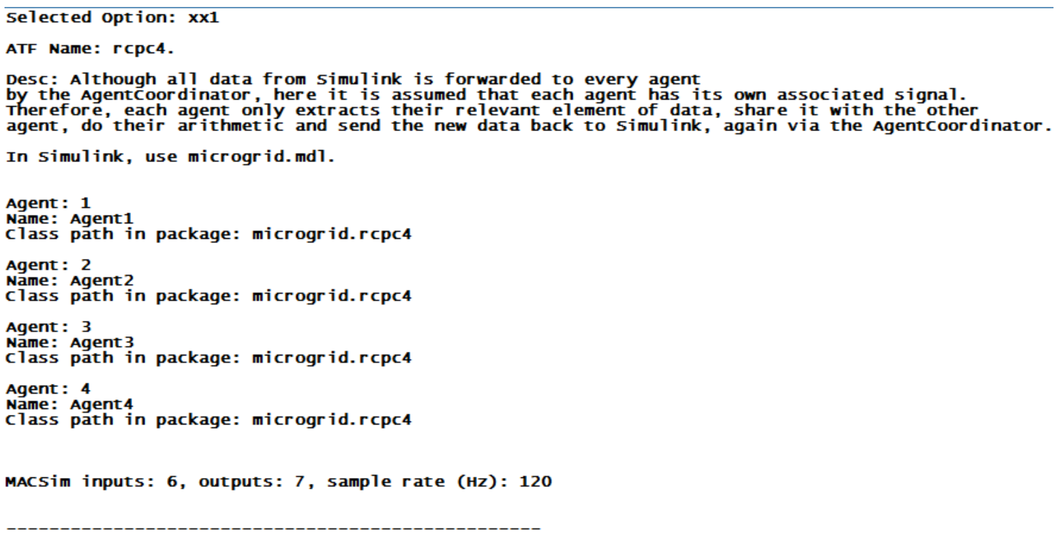


Figure 4.5. UML diagrams for some core Java classes including TimeStepData, AgentCoordinator, UsefulAgentMethods, Agent1, and Agent2

The ATF consists of all the agents that jointly operate on the data arriving from Simulink in order to accomplish some tasks. All of these agents will have setup() and takedown() functions and probably some behaviour functions. However, with the exception of communication protocols with the agent environment, the implementation of the functions will take on very different forms depending on the particular task the designer wishes them to achieve. A snapshot of an ATF file in the RCPC-8 control system is provided in Figure 4.6. The name of the task agents (Agent1, Agent2, Agent3, and Agent4), the number of input (e.g. i_1, i_2, i_{PL}) and output (e.g., x, y, z, u) signals and the frequency of running the RCPC-8 control platform (e.g., 120 Hz) are shown in this snapshot.

There can be any number of agents running concurrently and each has the ability to communicate with the other agents of the task force through AgentCoordinator using its associated *signal*$_i$ object. Therefore, each agent extracts only its relevant element of data, shares it with the subscribed agents, does its arithmetic, and sends the new data back to Simulink, again via the AgentCoordinator.



```
Selected Option: xx1

ATF Name: rcpc4.

Desc: Although all data from Simulink is forwarded to every agent
by the AgentCoordinator, here it is assumed that each agent has its own associated signal.
Therefore, each agent only extracts their relevant element of data, share it with the other
agent, do their arithmetic and send the new data back to Simulink, again via the AgentCoordinator.

In Simulink, use microgrid.mdl.


Agent: 1
Name: Agent1
Class path in package: microgrid.rcpc4

Agent: 2
Name: Agent2
Class path in package: microgrid.rcpc4

Agent: 3
Name: Agent3
Class path in package: microgrid.rcpc4

Agent: 4
Name: Agent4
Class path in package: microgrid.rcpc4


MACSim inputs: 6, outputs: 7, sample rate (Hz): 120

--------------------------------------------------
```

Figure 4.6. A Snapshot of an ATF File for the Agent1 Project Developed for Optimization of the Case Study

Figure 4.7 displays the steps for design and implementation of the RCPC-8 system that are taken in this research.

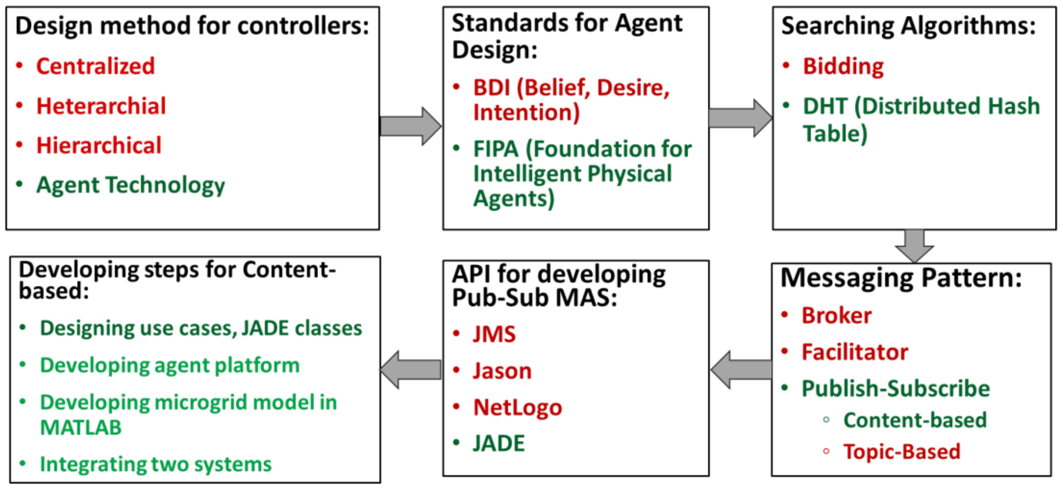

Design method for controllers:
Centralized
Heterarchial
Hierarchical
Agent Technology
Standards for Agent Design:
BDI (Belief, Desire, Intention)
FIPA (Foundation for Intelligent Physical Agents)
Searching Algorithms:
Bidding
DHT (Distributed Hash Table)
Messaging Pattern:
Broker
Facilitator
Publish-Subscribe
Content-based
Topic-Based
API for developing Pub-Sub MAS:
JMS
Jason
NetLogo
JADE
Developing steps for Content-based:
Designing use cases, JADE classes
Developing agent platform
Developing microgrid model in MATLAB
Integrating two systems

Figure 4.7. MAS Design Process

Chapter 5. Implementation of RCPC-8 Agent Platform on a Developed Simulator; Integrating RCPC-8 and MG-8

In this chapter, the steps are presented for developing an integrated RCPC-8 system based on a proposed algorithm, which is developed in section 2.2. It discusses one of the main contributions of this thesis, which addresses how several requirements can be met in a single platform design. In this chapter, the following actions are performed:

1. Implementing an agent-based platform using designed UML models for the publish-subscribe communication algorithm developed by JADE programming in Eclipse IDE
2. Developing a four-converter microgrid model on Simulink
3. Developing a real-time communication tool for integrating the hardware microgrid model in Simulink and multi-agent control systems. The communication tools should be able to run multiple agents simultaneously.

5.1. Implementing an Agent Platform based on a Publish-Subscribe method using JADE

Publisher and subscriber agent classes are implemented using JADE. The adopted communication paradigm is the asynchronous message passing. Receiver (subscriber) agents send messages to supplier (publisher) agents to request a variable. Each agent has a type of mailbox (the agent message queue) where the JADE runtime posts messages sent by other agents. Whenever a message is posted in the message queue, the receiving agent is notified (Figure 5.1). A receiver agent, which is previously subscribed for a particular content, activates an action method to start communication if there is any matching message, while ignoring all non-matching messages.

Figure 5.2 shows the work-flow diagrams of running agent platform by receiving trigger from Simulink model. Figure 5.3 displays the top-level flowchart of running the agent-based control system integrated to MATLAB, which is known as RCPC-8. Load values measured by the converter activate the agent model. As shown, the microgrid sends the load values to a S-Function block called MACSim (Multi-Agent Control for Simulink), which will be explained in section 5.2. This trigger initializes the process of optimization within the JADE platform. The frequency of running optimization processes in a JADE Platform is defined by a number between 1 and 120 cycles per seconds for the case study. This agent-based control system generates global optimized values.

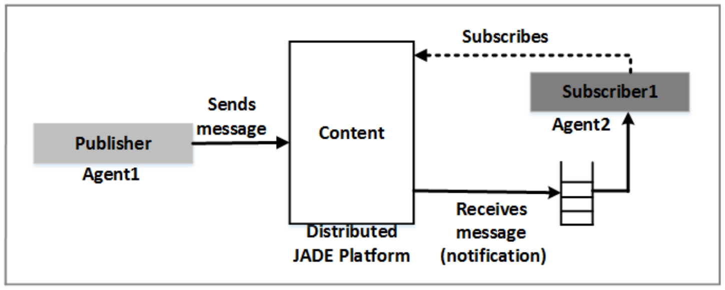
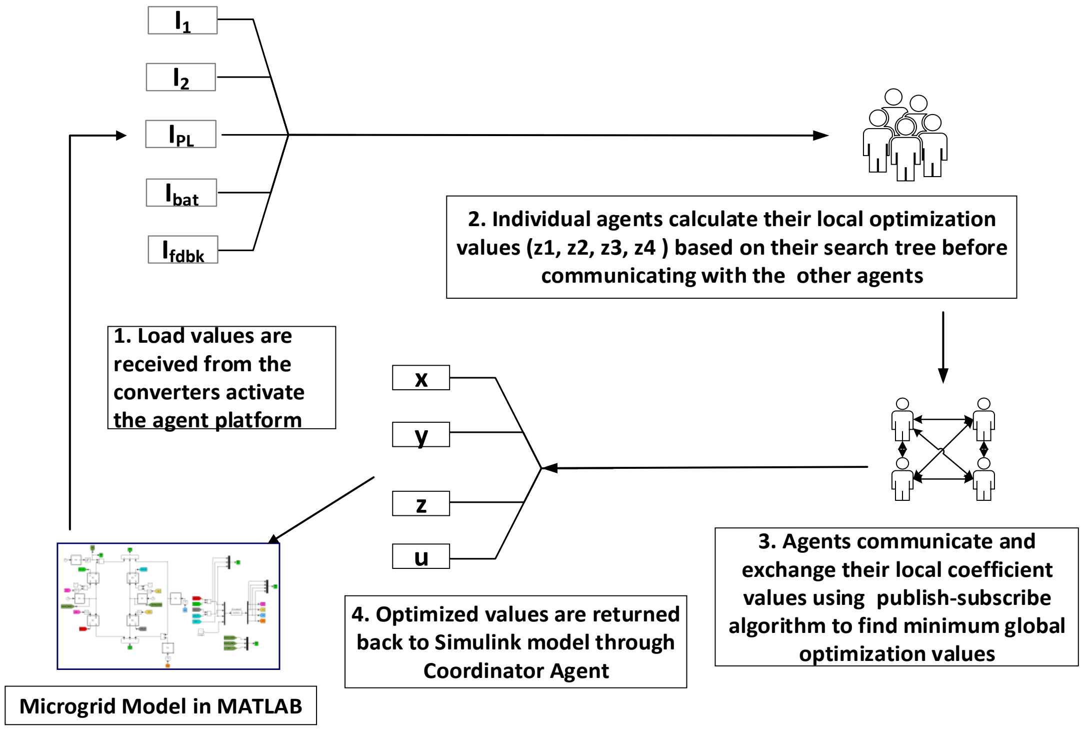


Figure 5.1. A JADE Asynchronous Message Passing Paradigm using the Content -based Publish-subscribe Design



I_1
I_2
I_{PL}
I_{bat}
I_{fdbk}
2. Individual agents calculate their local optimization values (z1, z2, z3, z4) based on their search tree before communicating with the other agents
1. Load values are received from the converters activate the agent platform
x
y
z
u
3. Agents communicate and exchange their local coefficient values using publish-subscribe algorithm to find minimum global optimization values
4. Optimized values are returned back to Simulink model through Coordinator Agent
Microgrid Model in MATLAB

Figure 5.2. Work-flow Diagrams of Running Agent Platform by Receiving Trigger from Simulink Model

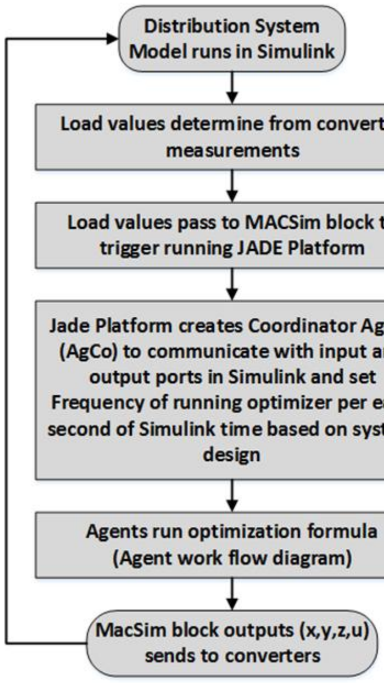

Distribution System Model runs in Simulink
Load values determine from converter measurements
Load values pass to MACSim block to trigger running JADE Platform
Jade Platform creates Coordinator Agent (AgCo) to communicate with input and output ports in Simulink and set Frequency of running optimizer per each second of Simulink time based on system design
Agents run optimization formula (Agent work flow diagram)
MacSim block outputs (x,y,z,u) sends to converters

Figure 5.3. Individual Agents (Ag#) Life Cycle during Establishing Communication and Data Transfer among Agents in the RCPC-8 System

Figure 5.4 displays the flowchart of the RCPC-8 control system emulated using JADE in Eclipse. Load values detected by the converter activate the agent model. Consequently, the JADE platform creates four individual agents called Ag1, Ag2, Ag3, and Ag4, upon trigger receipt. Since all of the agents run simultaneously, they concurrently extract z_i optimization values from input values (current values of Zone1, Zone2, pulsed load, ESS, and bus-tie that are represented as I_1, I_2, I_{PL}, bat_c, and fdbk respectively in Figure 5.4).

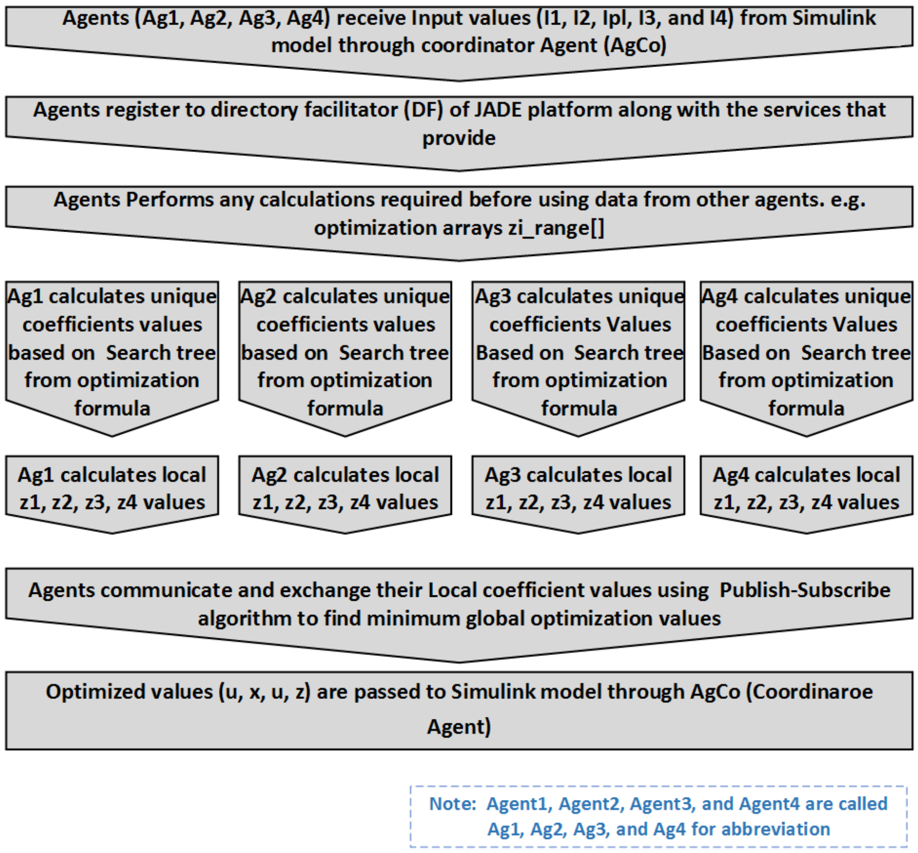

Agents (Ag1, Ag2, Ag3, Ag4) receive Input values (I1, I2, Ipl, I3, and I4) from Simulink model through coordinator Agent (AgCo)
Agents register to directory facilitator (DF) of JADE platform along with the services that provide
Agents Performs any calculations required before using data from other agents. e.g. optimization arrays zi_range[]
Ag1 calculates unique coefficients values based on Search tree from optimization formula
Ag2 calculates unique coefficients values based on Search tree from optimization formula
Ag3 calculates unique coefficients Values Based on Search tree from optimization formula
Ag4 calculates unique coefficients Values Based on Search tree from optimization formula
Ag1 calculates local z1, z2, z3, z4 values
Ag2 calculates local z1, z2, z3, z4 values
Ag3 calculates local z1, z2, z3, z4 values
Ag4 calculates local z1, z2, z3, z4 values
Agents communicate and exchange their Local coefficient values using Publish-Subscribe algorithm to find minimum global optimization values
Optimized values (u, x, u, z) are passed to Simulink model through AgCo (Coordinaroe Agent)
Note: Agent1, Agent2, Agent3, and Agent4 are called Ag1, Ag2, Ag3, and Ag4 for abbreviation

Figure 5.4. Running Optimization Algorithm on Four Agents Concurrently in the RCPC-8 System

The agents communicate and exchange data based on a publish-subscribe design. After each agent optimization routine has completed, agents with the minimum z_i values locate received data from peer agents. This agent calculates x, y, z, u values and sends them back to Simulink through a coordinator agent (AgCo). After receiving the confirmation of data delivery, each agent terminates and finishes its life cycle. These agents use individual search trees to optimize values of z_1, z_2, z_3, and z_4. The mathematical definition of a publish-subscribe design for communication and exchanging data among agents is explained in (5.1) – (5.8).

In addition, the coordinator agent is developed to facilitate communication among Simulink ports and the agent platform. Optimized values then return to the converters through Simulink ports. All of the routing tables are located in the coordinator agent, which communicates with the other agents at the beginning and end of the optimization processes. Figure 5.5 illustrates a snapshot of a message exchange among agents. Each of the four aforementioned agents registers to the DF agent of the JADE platform to receive its agent number and locate its local data provider. An individual agent performs all of the required calculation before using data from other agents. Each agent then sets the data structure information in to be passed to the subscribed agents using an associated *signal*$_i$ attribute.

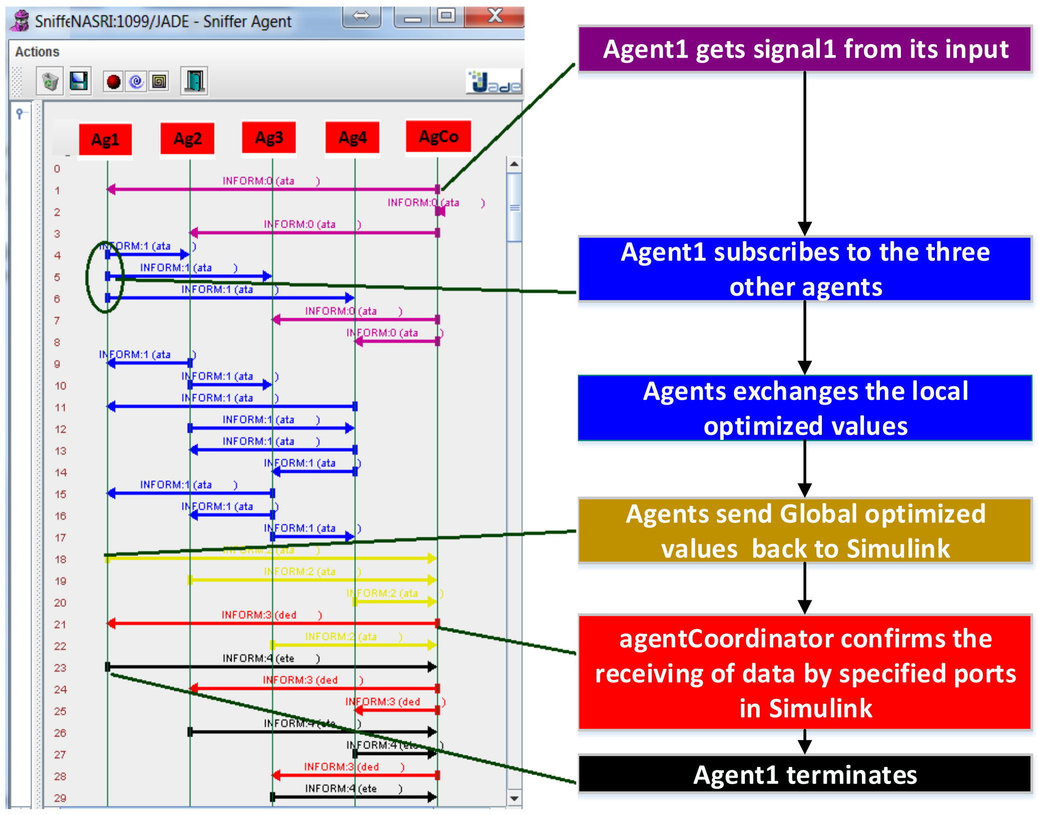

SniffeNASRI:1099/JADE - Sniffer Agent
Actions
Ag1
Ag2
Ag3
Ag4
AgCo
Agent1 gets signal1 from its input
Agent1 subscribes to the three other agents
Agents exchanges the local optimized values
Agents send Global optimized values back to Simulink
agentCoordinator confirms the receiving of data by specified ports in Simulink
Agent1 terminates

Figure 5.5. A Snapshot of a Sniffer Showing Communication between Agents with the Application of Publish-Subscribe Architecture in a JADE Platform

5.2. Bridging the Gap: MACSim Block

Whilst Simulink is very effective for carrying out simulations, it falls short of offering the tools necessary to set up an agent platform. One very useful aspect of Simulink, however, is that it provides a work-around for adding functionality in the form of S-functions. These allow programs to be written in other languages, particularly C, that can be encapsulated in the Simulink environment and then used where desired, running in their native language [98].

Despite this prospect of a solution, where the agents could be created through C++ or Java code in one of these functions and run in Simulink, there is a further setback. S-functions are unable to handle multiple threads of execution and they become unstable if several processes run concurrently inside Simulink. Unfortunately, this functional property is essential for a multi-agent system. To overcome this problem, a program called MACSim is used and customized so that it still utilizes the S-function ability of Simulink, but only as a gateway to pass data to a completely separate program where multi-agent development is possible. MACSim, which was originally developed by Mendham [99], provides an extension enabling an interface between Simulink and JADE for modelling hardware run by software agents.

MACSim, or the Multi-Agent Control for the Simulink program, was purposely developed as a medium through which a program for implementing agent designs developed in C/C++ or Java might pass data to and from Simulink [100]. Although MACSim is written primarily in C++, it includes a wrapper to enable interaction with Java programs. MACSim has a client-server architecture, where the client part is embedded in Simulink through an S-function, and the server code is then incorporated in the separate program as indicated in Figure 5.6. In addition, as shown, publish-subscribe architecture is used in this thesis for the design of a JADE control platform.

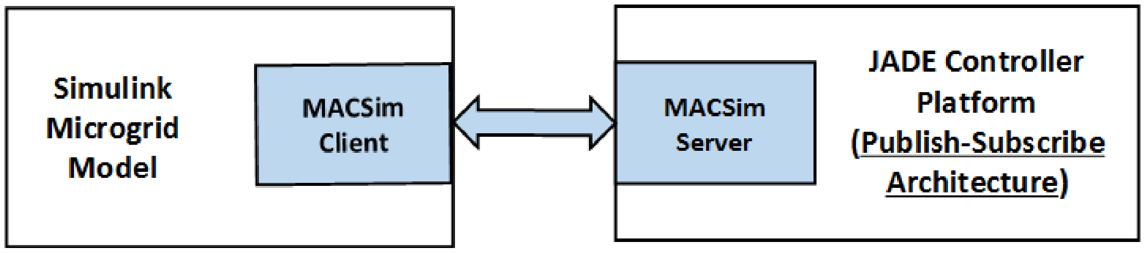


Figure 5.6. Structure of MACSim for RCPC-8 [112]

The communication between the client and server is then performed through the use of pipes in Windows. Use of MACSim circumvents the multi-threading issue because a separate program can now be used with protocols in place to ensure synchronicity if so desired. Later on, Robinson created MACSimjx, which is compatible with a JADE platform. MACSimjx and four JADE libraries are installed on a JADE platform to facilitate creating agents' classes and connecting them to the MACSim S-function on the Simulink model [101]. MACSimJX provides MATLAB models developed in Simulink with access to JADE, a powerful development environment for modelling MAS. In this thesis, MACSimJX is used for developing agent platform and, for simplicity, is called MACSim.

One of the major contributions of this thesis is the development of a multi-agent controller on the server side of MACSim to allow the JADE controller to interact with the Simulink model in a real-time matter. The JADE template in this thesis is made of two significant sections: 1) Agent Environment, which is the extension of the MACSim server and is slightly modified from the original version [102] developed by Robinson at the University of York; 2) A multi-agent control system that is developed for operating on data arriving from Simulink through the Agent Environment. The second part of MACSim developed in JADE is described in the following section.

5.3. Designing an Integrated Testbed for Implementation in the Case Study

In addition to the time limitation, the RCPC-8 system has other constraints such as a maximum number of concurrent running tasks and asynchronous running speeds between hardware and software platforms. Hardware is a DC microgrid including eight converters that are simulated in MATLAB, and a software platform is an agent-based publish-subscribe control system implemented using a JADE platform. The two parts are connected to each other via MACSim block. Figure 5.7 illustrates a high level design of the RCPC-8 system.

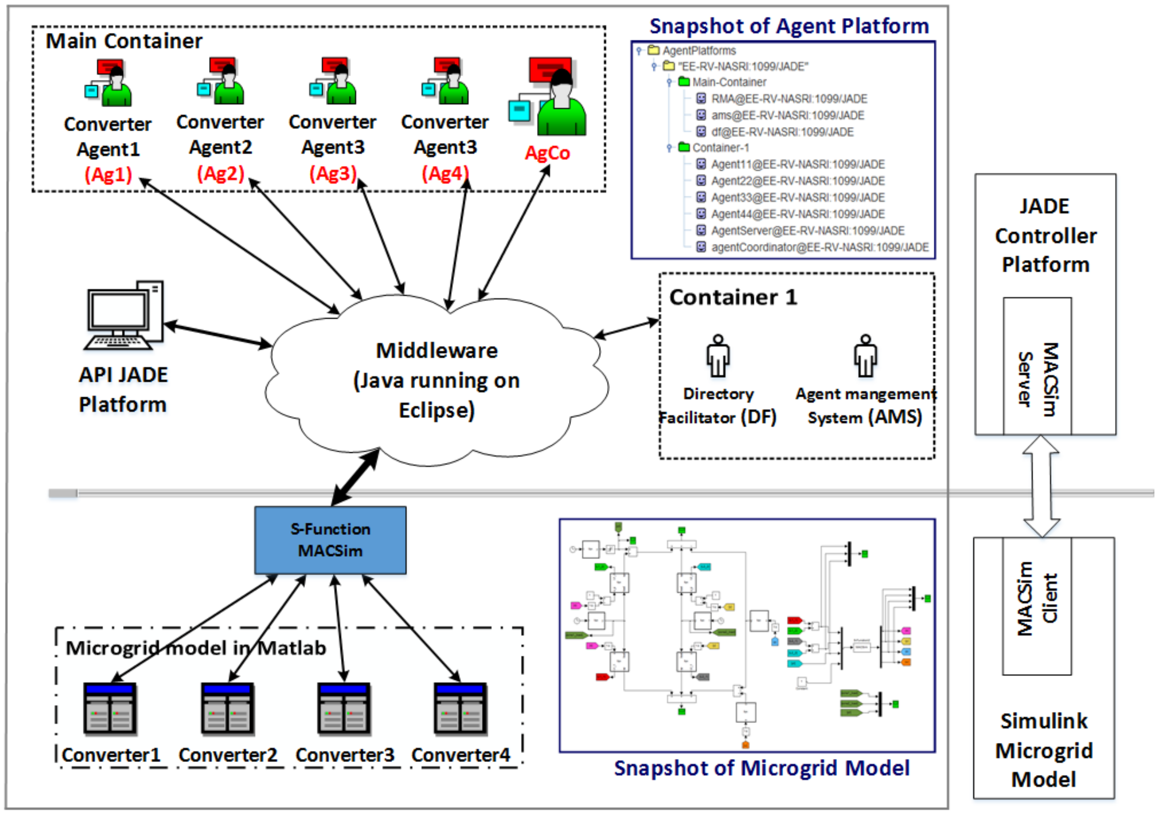

Main Container
Converter Agent1 (Ag1)
Converter Agent2 (Ag2)
Converter Agent3 (Ag3)
Converter Agent3 (Ag4)
AgCo
Snapshot of Agent Platform
AgentPlatforms
"EE-RV-NASRI:1099/JADE"
Main-Container
RMA@EE-RV-NASRI:1099/JADE
ams@EE-RV-NASRI:1099/JADE
df@EE-RV-NASRI:1099/JADE
Container-1
Agent11@EE-RV-NASRI:1099/JADE
Agent22@EE-RV-NASRI:1099/JADE
Agent33@EE-RV-NASRI:1099/JADE
Agent44@EE-RV-NASRI:1099/JADE
AgentServer@EE-RV-NASRI:1099/JADE
agentCoordinator@EE-RV-NASRI:1099/JADE
JADE Controller Platform
MACSim Server
API JADE Platform
Middleware (Java running on Eclipse)
Container 1
Directory Facilitator (DF)
Agent mangement System (AMS)
S-Function MACSim
Microgrid model in Matlab
Converter1
Converter2
Converter3
Converter4
Snapshot of Microgrid Model
MACSim Client
Simulink Microgrid Model

Figure 5.7. An integrated Agent-based System including a Matlab Model in the Lower Section and a JADE Platform in the Upper Section, Joint using MACSim Toolbox

Eight converters are individually assigned to converter agents that communicate through the JADE API platform. Converter agents communicate with each other inside main containers based on the system design. Regarding the FIPA standard definition in chapter 1, three other agents including DF, AMS, and ACC run in the main container. DF provides a directory that announces which agents are available on the platform. AMS is the only agent that is able to create and destroy other agents, destroy containers, and stop the platform.

5.4. Modeling a Microgrid with Eight Converters in MATLAB/Simulink

In this section, the implementation results obtained from the JADE platform are presented. To be able to perform experiments on a microgrid model, an agent environment was also developed, permitting experiments with up to 100 converters. All simulations were performed on an Intel® Core™ i7 processor (860 2.80 GHz CPU) with 16GBytes of main memory, running Windows 7 64-bits Enterprise. MACSim provides an extension enabling an interface between Simulink and JADE for modelling hardware run by software agents originally developed for MATLAB 2010 (32 bits), and JDK 32-bits. The JADE controller was configured to run in a single Java Virtual Machine. An agent-based control platform critically reduces convergence time among converters compared to the classic control model running on Matlab and developing the same algorithms.

The effectiveness of the distributed optimization algorithm and the proposed agent-based control method were verified by a proof-of-concept testbed and pilot implementations. The microgrid model shown in Figure 5.8, as described in Chapter 3, has two zones of utility loads and two power conversion modules (PCMs) in each zone, sharing the zonal load demand. Sharing of the zonal load may vary from 0% to 100% depending on the controller decision. The load center PCMs are assumed to be unidirectional. The Pulsed Load is one of the highest priority loads of the microgrid. Two main buses from the backbone of the microgrid are connected by two cross-tie disconnects. These disconnects are used to connect the two bus bars, control flow of inter-bus energy, maintain voltage levels and disconnect them as necessary.

Based on the optimization method in section 2.2, transmission loss of the sample microgrid can be simplified using a MIQP algorithm using the following cost function:

$$P_t = z_1^2 + z_2^2 + z_3^2 + z_4^2 + Const \tag{5.9}$$

where in the 'z_i' domain objective function, losses due to each variable form the shape of a parabola which gives a clear indication of the optimal region. Each 'z_i' value in (5.9) is calculated by an associated agent which is using x, y, z, and u parameters. Each agent can solve its portion independently without overlap. At the end of the computation, all of the agents concurrently update their load level using calculated optimal values. Load changes and corresponding results of the dynamic optimization of the converter set point is shown in Figures 5.8, and 5.9.

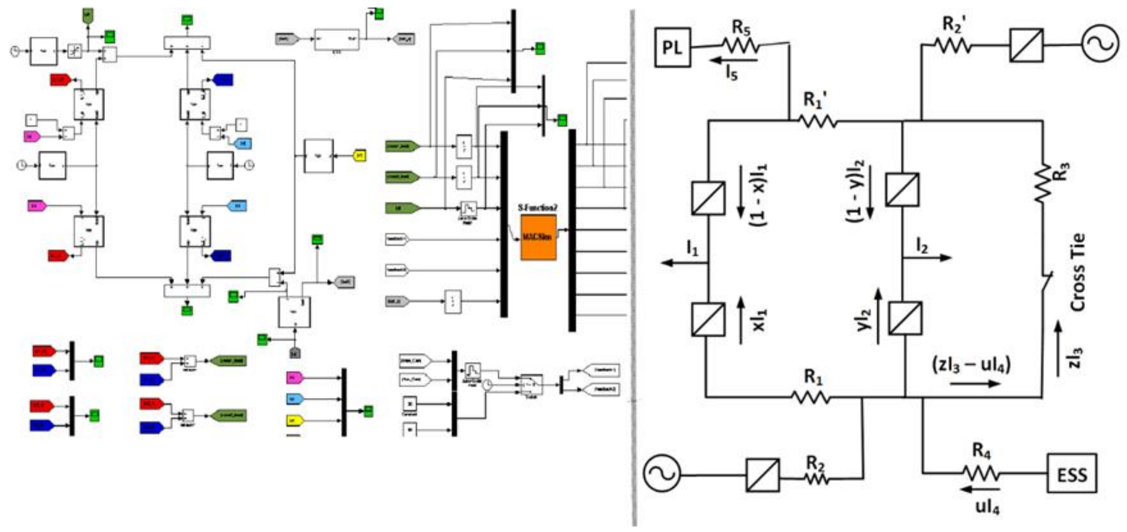


Figure 5.8. A Snapshot of a Shipboard DC Microgrid Modeled in Simulink

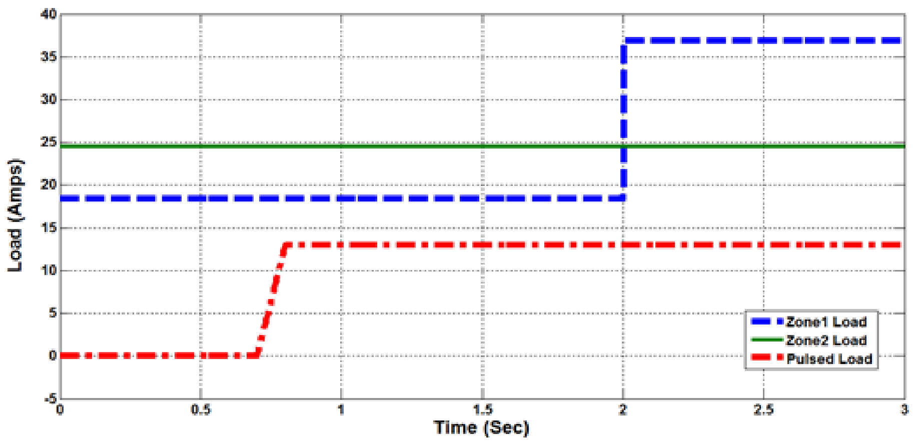


Figure 5.9. Load Changes in the Shipboard System

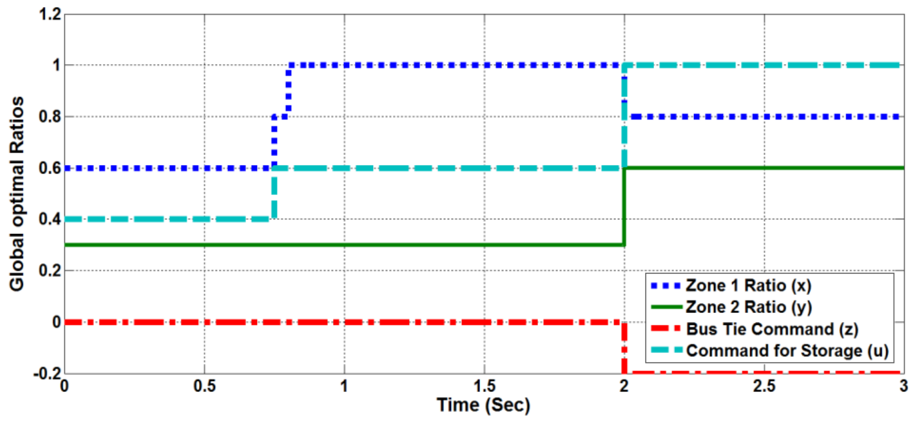


Figure 5.10. Converter Operating Points Dictated by the Load Changes

Figure 5.9 displays the changes of load levels in the microgrid model that are sensed by the agent control system during the time period of the running system simulation. Simulation results of load ratio adjustments are presented in Figure 5.10. As explained in Section 2.2, variables x, y, z, and u represent Zone1 ratio, Zone2 ratio, bus-tie situation, and command storage respectively that follow input load change values shown in Figure 5.9. The pulsed load storage system is charged rapidly at time t=0.7sec. Consequently, in Figure 5.10, at the same time x increases from 0.6 to 1.0, so one converter carries 100% of the zonal load and the other one runs at no load (0% load). ESS increases by 20% to maintain the load balance and provides the extra required load of Zone1. At time t=2sec when Zone1 load changes, all system configuration changes. To ensure maximum efficiency, the Zone1 sharing variable decreases from 1 to 0.8, the Zone2 sharing variable increases twice from 0.3 to 0.6 to flow the changes in x values, the ESS increases its supply to its maximum limit of 1.0, and the Starboard side bus provides 20% of its maximum allowable inter-bus energy flow to the Port side bus.

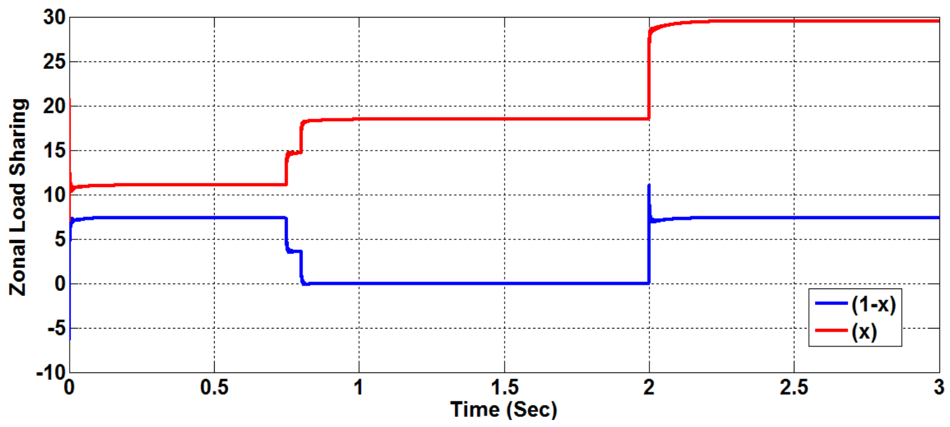


Figure 5.11. Load Sharing Measured in Zone 1

Sharing of load between Zone1 and Zone2 converters is shown in Figures 5.11 and 5.12. As is seen, sharing the variable of Zone1 and Zone2 carries (x)% and (y)% of the zonal load respectively and is illustrated with red lines diagrams. As the value of x increases in Figure 5.11, the corresponding change in load sharing is seen in Figure 5.12. At around 0.8 sec when x becomes 1, one converter carries 100% of the Zone1 load and the other one runs at no load (0% load). Zone1 load changes at t=2 sec, so does the ratio x. Similar change in load sharing is observed in Figure 5.12. Then, the other converter carries (1-x)% and (1-y)% of the load sharing and is presented with blue lines. Zonal voltages are shown in Figure 5.13. These voltages are maintained at 400V DC. It is seen that some minor shifts are occurred due to a change in loads and operating points, which are quickly recovered by the controllers.

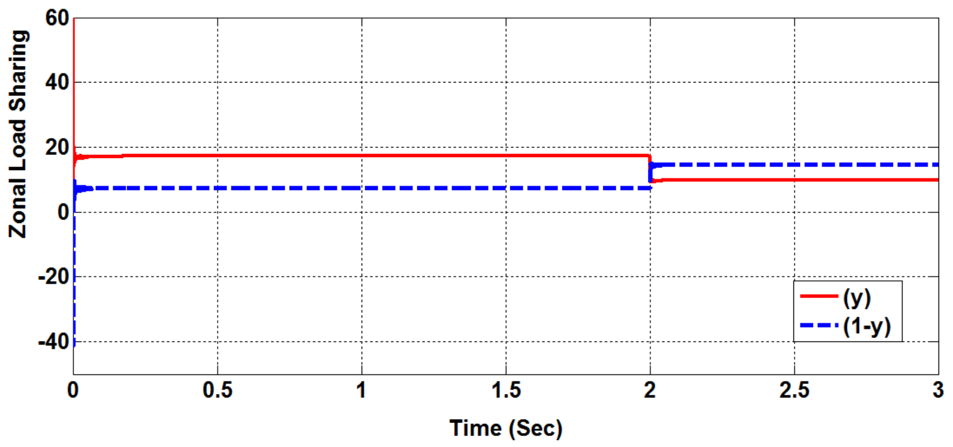


Figure 5.12. Load Sharing Measured in Zone 2

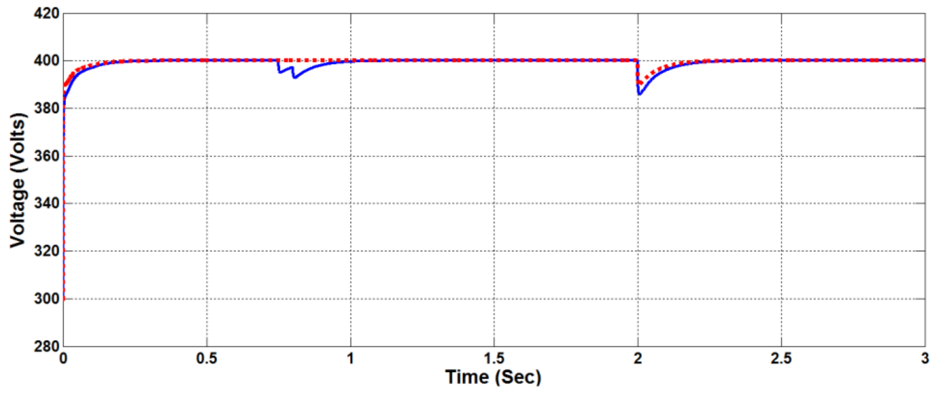


Figure 5.13. Zonal Voltages

5.5. Analysis of Simulation Results

Similar to chapter 5, the simplified cost function in (5.1) based on MIQP is used for calculating transmission loss of the sample microgrid. Hossain and Ginn presented the results of simulation using the same optimization algorithm applying a centralized control system developed in MATLAB [103]. The results are shown in Figure 5.14 and Figure 5.15, which are exactly the same as Figure 5.19 and Figure 5.10 respectively.

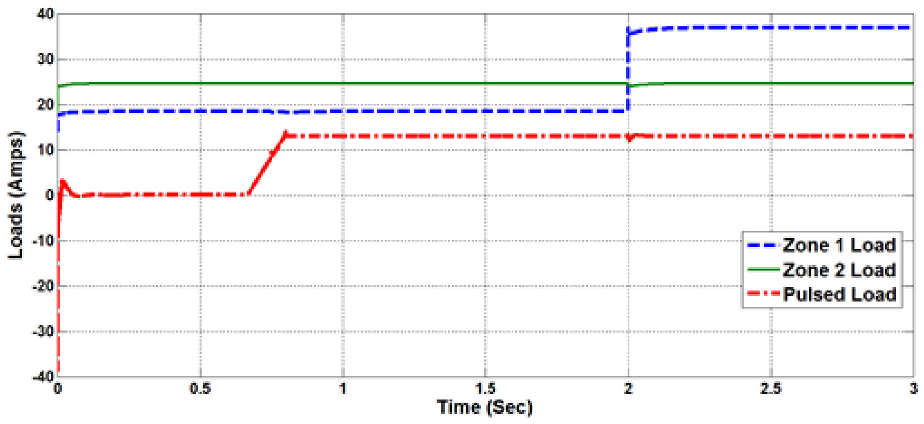


Figure 5.14. Load Changes in the Shipboard System

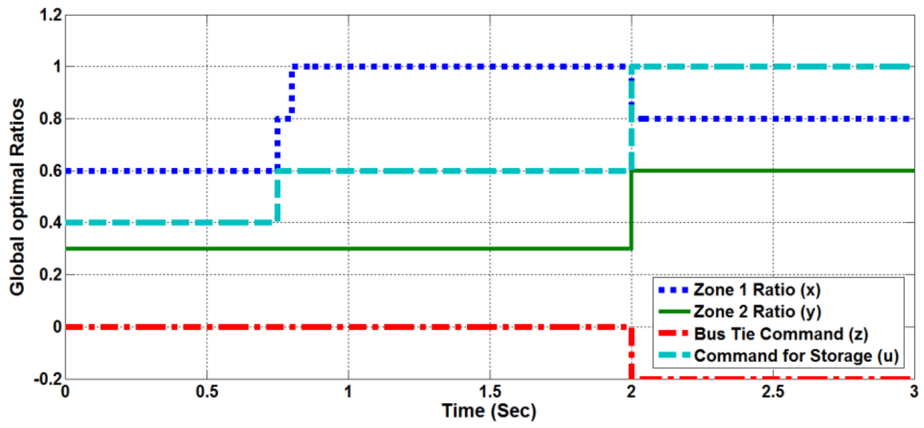


Figure 5.15. Converter Operating Points Dictated by a Centralized Control System Developed in MATLAB

5.6. Time Saving

A comparison between the results of running an optimization algorithm in MATLAB with and without application of integrated agent-based control systems shows that the load sharing ratios for Zone1 and Zone2 are related and adjust their values based on each other’s changes. However, Figure 5.16 illustrates the critical time difference between adjusting the ratio of load sharing in controllers that were developed by MATLAB, with and without using the JADE platform.

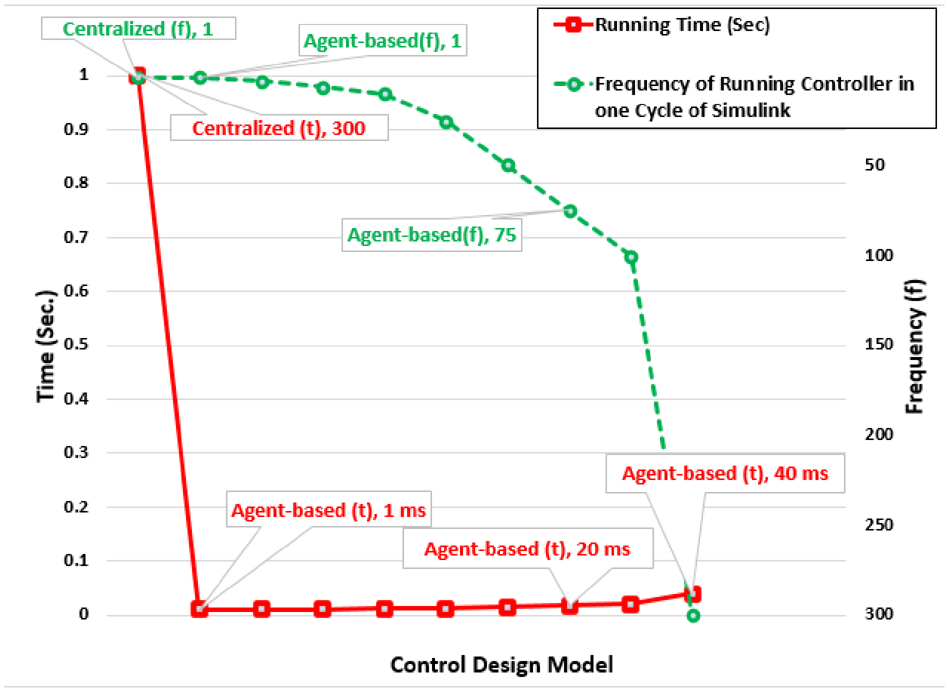


Figure 5.16. Results of Time Analysis During Coordination Time; Agent-based(f), and Agent-based(t) Stand for Frequency and Time of Running Agent-based Control Model, Centralized(f), and Centralized(t) Stand for Frequency and Time of Running Centralized(MATLAB) Control Model

For the case study, while the coordination time in the first model takes about five minutes (1000 milliseconds), it takes less than three seconds for the later one, which is only 1% of the time consumed in the centralized controlled model. More importantly, the optimal frequency value of running the agent controller within the Simulink model is set for 120 cycles per second and can be modified based on system requirements. This means that the RCPC-8 real-time control system is able to check the changes in less than one millisecond by keeping the consistency of the upper time limit of the optimization process.

Figure 5.17 compares the number of message exchanges that resulted from the simulation in Figure 5.5 with the numerical analysis data extracted from Figure 2.10. As mentioned in Section 2.4, the numerical analysis graph is calculated based on the publish-subscribe architecture. Also, as explained in the introduction of chapter 4, the number of message exchanges for numerical analysis calculated from (5.10) for the RCPC-8 system by definition of n and r values that are based on the system design [104], [105]:

$$Number\ of\ message\ exchanges\ from\ the\ numerical\ analysis = r\ log_2\, n \qquad (5.10)$$

$$\begin{cases} n\ is\ the\ number\ of\ nodes\ (\ equal\ to\ the\ number\ of\ agents\ ):\ 3 \le n \le 8 \\ r\ is\ the\ average\ length\ of\ string\ values : r =\ 44 \end{cases}$$

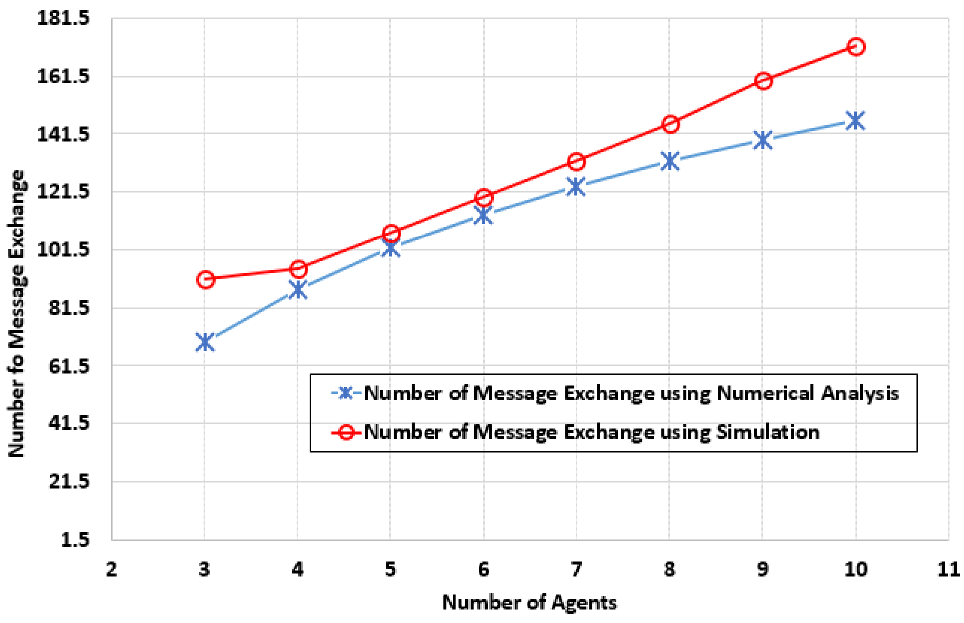


Figure 5.17. Comparison between Numbers of Message Exchanges for the Publish-Subscribe Method that Resulted from Numerical Analysis (Figure 2.10) and Simulation (Figure 5.5)

The blue line in Figure 5.18 displays the number of message exchanges for a variation between 3 to 8 agents using numerical analysis. The red line displays the number of message exchanges extracted from the simulation results. As shown, the simulation results confirm the numerical analysis outcomes; the most optimal number of agents is from 4 to 7, which is chosen in our design for eight converters. When the number of agents increases, they run more parallel treads which are expected to improve the system efficacy. However, considering the amount of overhead in message exchanges, it does not practically advance the efficacy of the system. In this thesis, as explained in chapters 4 and 5, five agents—four Converter Agents(CAgs) along with one Coordinator Agent(CoAg)—are designed and implemented for coordination control of the sample microgrid.

Figure 5.18 displays a comparison among the time period of adjusting the optimization values after applying load changes to the system. As table 5.1 illustrates, with a frequency of 120, the coordination time among agents is 0.0083 seconds. The results are reasonable by considering the sampling time rate for the Simulink model that is defined as 120 cycles per second. Although the frequency of running the JADE platform can be increased, the accuracy of running the controller will not exceed the MATLAB sampling rate.

Table 5.1. Relation between frequency of running agent-based control model and time-delay of adjusting load changes

Frequency (Hz)	Load Changes (Applied at T=2 Sec)	Load Changes (Applied at T=1 Sec)	Difference for Time Adjustment
1	3	2	1
10	2.1	1.1	0.1
15	2.0667	1.0667	0.0667
20	2.0333	1.0333	0.0333
30	2.025	1.025	0.025
50	2.0167	1.0167	0.0167
100	2.0167	1.0167	0.0167
130	2.0083	1.0083	0.0083
150	2.0083	1.0083	0.0083

Frequency (Hz)	Load Changes (Applied at T=2 Sec)	Load Changes (Applied at T=1 Sec)	Difference for Time Adjustment
240	2.0083	1.0083	0.0083

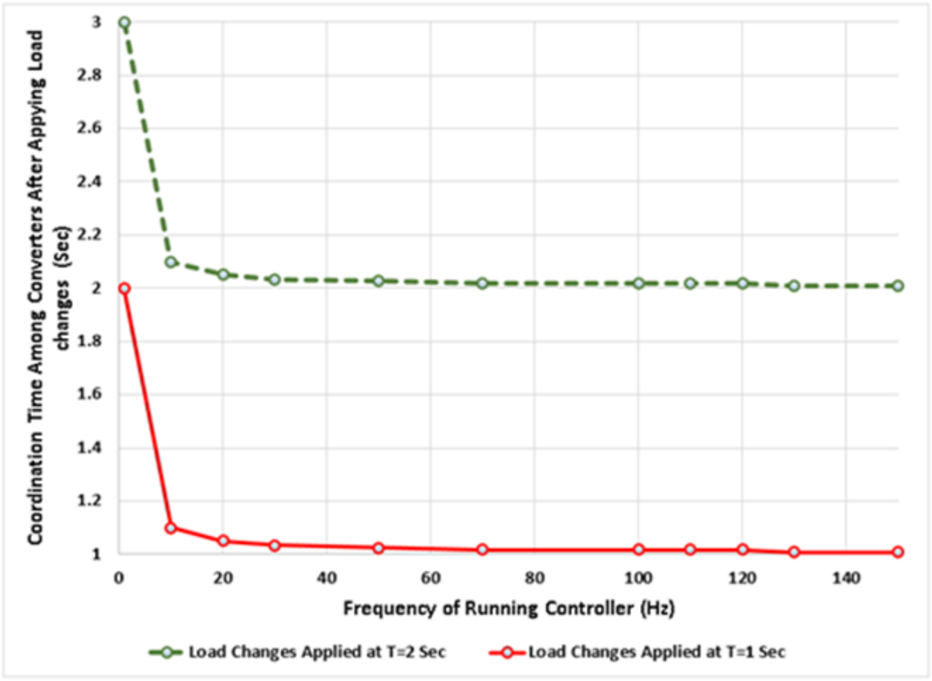


Figure 5.18. Comparison among Adjusting Load Level of Converters after Applying Changes in Different Times

Different microgrids have different operational requirements and goals. Agent technologies are deployed to meet these requirements based on system design. Microgrids can have unidirectional or bidirectional energy flow, and require soft real-time or hard real-time control systems. Hermann Kopetz defines [106] a real-time system as “a computer system where the correctness of the system behavior depends not only on the logical results of the computations, but also on the physical time when these results are produced. System behavior means the sequence of outputs in time of running a system." A real-time power system requires the real-time data provision and communication between the power system control and operation framework [107]. A system is working in hard real-time if its operation depends on both the logic correctness and its performance time. It means missing a deadline causes system failure. Soft real-time tolerates delay but operates with reduced service quality. Fast power system monitoring and regulation are hard real-time operations because their delay can cause power system dynamic stability problems or failures in power system operation [108]. The system goals for both test systems used in this research are selected such that they are soft real-time because delay can increase the loss of system and affect negatively their functionality, but it doesn’t cause failure of operation.

Chapter 6. Developing an Agent Platform for Adaptive Volt-VAR Optimization (VVO) as Third Case Study

The current generation of electricity systems are expected to have capabilities such as DMS and EMS to improve their efficiency. Due to the effects of load profiles on the quality of delivered energy to customers, new technologies need to be developed for monitoring and control of reactive power and energy losses in distribution systems. One of the main techniques traditionally employed to reduce losses in distribution feeders is Volt-VAR Optimization (VVO) [109], [110], [111]. VVO is an advanced method that optimizes voltage and/or reactive power (VAR) of a distribution network based on predetermined aggregated feeder load profile. This is normally done using Load Tap-Changers (LTCs), Voltage Regulators (VRs), Capacitor Banks (CBs) and other existing Volt-VAR control devices in distribution substations and/or distribution feeders [112]. The configuration of traditional VVO systems is essentially based on offline techniques while only a few studies have considered VVO or CVR for real-time applications. Conservation Voltage Reduction (CVR) is also used to save delivered power, while keeping the delivered voltage within the acceptable American National Standard Institute (ANSI) prescribed range [113], [114]. Attempts have been made to employ IAs for real-time command and control capabilities required to transform conventional static VVO systems into real-time, adaptive, and dynamic VVO solutions.

Recently, new approaches towards VVO have been studied by research groups [115] and some utilities' companies such as BC Hydro [116] and Hydro-Quebec [117]. Nevertheless, most such works are still in their initial phases of development. Given the need to incrementally modernize the assets of distribution substations, studies of real-time adaptive VVO have become critically important. The approach in that regard is to develop new VVO optimization algorithms based on load profiles calculated using real-time measurements of service quality at the point of delivery to customers. Smart grids have been objected to use a combination of centralized and decentralized (distributed) control systems [118]. Centralized control systems have the best performance for small scale power networks and delivering power in one direction (i.e., from substation to loads). This section proposes an MAS to optimize an adaptive real-time VVO algorithm and achieve higher degrees of efficiency and reliability. Figure 6.1 depicts the IA system for VVO application. Real-time optimization and control in this system is performed by VVO Engine (VVOE).

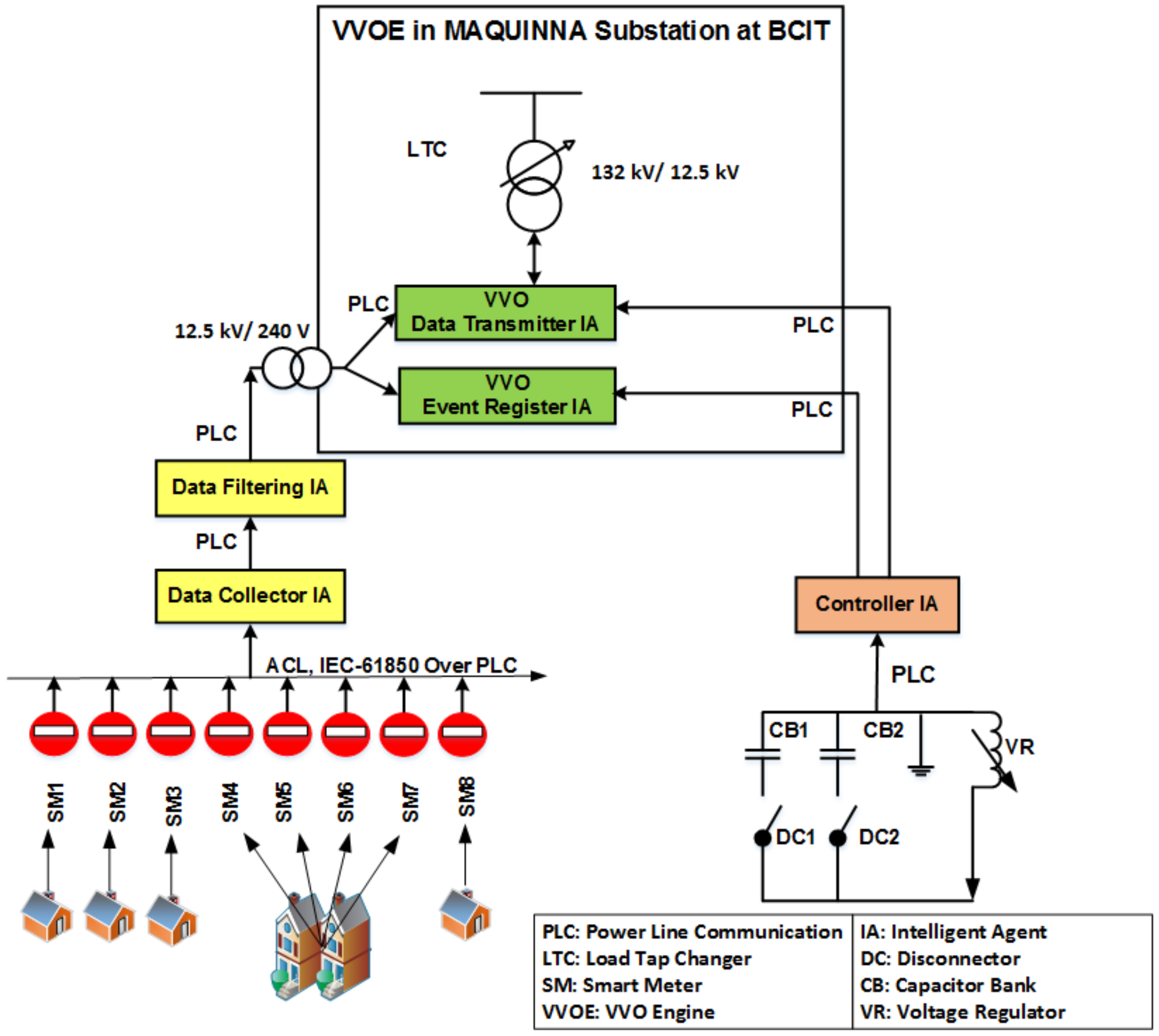


Figure 6.1. Agent-based VVO Structure in a Distribution Network

6.1.1. MAS Proposed for VVO

The proposed MAS employs capabilities of other software applications (e.g., interaction with a multi-task software) to enable the optimization of Volt-VAR values [119]. A supervisory agent for communication between agents using MPI (Message Passing Interface) was defined for integrated control and asset management of petroleum production facilities [120]. As a constituent component of the work in this thesis, a control system for Volt-VAR optimization of substation automation is designed using an agent technology based on IEC 61850's Goose Messaging protocol.

As Table 2.2 lists, IAs are classified into five types based on their profile: 1) a VVO data transmitter IA which is responsible for message transactions between the VVO server inside the substation, controller IA, and data filtering IA; 2) a VVO event receiver IA which captures alarm messages from all other IAs; 3) a data collector IA which collects required data types from Smart Meters (SMs) in defined rolling intervals through a Modbus protocol; 4) a data filtering IA which processes, filters, and transfers data to VVO IAs; and 5) a controller IA which is responsible for applying optimal reconfigurations to Volt-VAR control devices, analyzes results, and sends feedback to VVO IAs. The required data captured by IAs include, but are not limited to, active power, reactive power, and voltage. The IAs communicate with each other using an ACL which is compatible with the IEC 61850 standard.

Upon completion of optimum settings for its assets, Multi-agent control system reconfigures the network by sending configuration controls to Volt-VAR control devices such as CBs, VRs, and LTCs. Due to limited bandwidth and higher efficiency requirements, message packets are loaded with an optimum amount of data display agents such as voltage, active, reactive power, etc. Moreover, the other parameters are calculated in the VVO IA inside the substation server. In addition, each parameter in a Smart Meter has a unique code (similar to a MAC Address) that is registered in the network after connection and is signed out after exit. More details about this system can be found in the two papers published on this topic [121], [122] .

Table 6.1. **Capabilities and data parameters to be controlled with each Agent; Active power (P), Reactive power (Q), Voltage (V), Current (I), and Power factor (PF).**

Agent's Name	Capabilities	Data Parameters	Associated Agent Layer
VVO Data Transmitter	Receiving data, sending commands, optimizing Volt-VAR	P, Q, V, I, PF	Planning
VVO Event Receiver	Receiving alarms, optimizing Volt-VAR	P, Q, V, I, PF	Reactive
Smart Meter/ Data Collector	Receiving data, alarms and commands	P, Q, V	Reactive
Data Filtering	Receiving and sending data, processing data, generating and sending alarms, applying ANSI band, sending event, receiving command	P, Q, V	Coordinator
Controller	Receiving command and data, processing data, applying control commands and sending data and alarm	V, I	Coordinator & Reactive

Chapter 7. Conclusions and Future work

In order to optimize various control parameters in an autonomous microgrid, controllable devices, such as power sources, should be closely coordinated using a compatible control technology. This work proposed the application of multi-agent systems for cooperative control in microgrids. Two control agent platforms were developed in this thesis to apply the optimization functions to two separate test systems. The first test system was a connected AC microgrid which used an agent platform for Volt-VAR Optimization (VVO) among power devices, while the second test system was an islanded DC shipboard microgrid which deployed another agent platform to coordinate power flow among power electronic converters. Considering different requirements for the design of the aforementioned agent-based control models, different agent technologies are applied for two individual test systems.

The VVO test system was a small-sized grid-connected AC microgrid including eight smart meters and other power devices such as capacitor banks. The system was geographically distributed along the substation and customers' houses and used for optimizing power losses where voltage and reactive power were defined as controllable parameters. In the other test system, a DC shipboard system was defined as a medium-sized islanded microgrid including tens of converters. The agent-based controller successfully determined the optimum power sharing among converters to ensure minimum transmission and distribution loss while enforcing constraints such as the sources' capacity limits. The two test systems had very different time requirements. The VVO model had a broad time range of 1-15 minutes while the islanded DC microgrid was considered as a soft real-time system with a deterministic time frame of a few tens of milliseconds. Different timing requirements led to the application of different agent layers in the optimization processes.

It was mentioned in section 1.6 that agents are located in the reactive layer if they are pre-programmed to do certain tasks. Also, they are placed in the planning layer if they are context dependent, cooperating in their local tasks and competing with similar agents in other nodes in pursuing global goals. When real-time tasks are required, they are usually handled using a reactive layer's agent because they are closely associated to the power electronic devices such as converters. In this thesis, while all three agent layers of planning, coordinator, and reactive are deployed for the VVO test system, the islanded DC microgrid only used coordinator and reactive layers. Consequently, while publish-subscribe agent technology was extracted as the most efficient agent platform for the DC shipboard microgrid, the VVO agent-based control platform was designed and developed using the facilitator model which meets all the requirements of that system. A summary of comparison among two test systems is explained in Table 2.1.

For the first test system, an agent platform was developed and employed for VVO in a connected AC microgrid. The advantages of using MAS were stated and the design methodology, architecture, and interactions among the five employed agents were discussed. This agent platform was tasked with processing smart metering data, determining probable events which have produced that date, and communicating their findings with the VVO Intelligent Agent, which in turn can determine the new configuration settings for the VVO components in the system.

In the path to extract the most efficient agent-based distributed control technology for the second test system, a medium-sized DC shipboard microgrid with 32 electrical converters was used for numerical analysis. Different agent technologies including Belief, Desire, Intention, and Holonic were presented and analyzed numerically using an agent platform (RCPC-32). Furthermore, the management of adding and deleting converters based on Distributed Hash Table (DHT) infrastructure displayed a consistent and high level of efficiency for the test system. Therefore, a combination of the publish-subscribe agent platform and the DHT searching algorithm was opted as the most efficient, scalable technology for real-time coordination of power converters in the defined microgrid. It was shown that an increase in the number of power converters from 1 to 50 did not visibly affect the upper time limit of message exchange.

Another main contribution of this thesis has been to develop an integrated system for checking the optimization algorithm. It includes a microgrid model, agent-based control platform and integration block which had to be able to run multiple optimization processes simultaneously. A simplified version of a DC shipboard microgrid, including eight converters, was developed and simulated in MATLAB/Simulink as the case study (MG-8). Simulink is usually used for modeling optimization algorithms in microgrids, but it cannot implement the multithreaded agent architectures using agent development tools and standards that are available in other languages such as JAVA. Therefore, Simulink was coupled with other tools to realize an overall agent simulation platform.

A Java Agent DEvelopment Framework (JADE) was chosen for developing an agent platform because it provides a set of internationally recognised standards defined in FIPA. JADE is particularly well suited for building multi-agent systems and can address the weakness in Simulink, but the two programs must be synchronously linked in order to facilitate interaction among their combined attributes through Java classes. The content-based publish-subscribe method has been modeled for this system using the JADE Platform. The content is defined for each publisher and subscriber based on the required data for running an optimization formula by each agent. To aid the design process for decentralized data fusion using agents, an interface called MACSim has been customized, re-programmed, and used that integrates Simulink and JADE. Thus an integrated multi-agent software system is available for the development, testing and analysis of the shipboard case study.

In the publish-subscribe model, the number of message exchanges extracted from the simulation agree with the analytical results. The agent platform deployed distributed optimal power-sharing algorithms while maintaining a deterministic time frame of a few tens of milliseconds for a system with tens of converters. In addition, there was no limitation in the number of events which might happen concurrently. Results of implementing the agent-based publish-subscribe control system using JADE were illustrated in this thesis.

In summary, the application of the publish-subscribe agent technology led to balancing load sharing upon the changes in the system load flow. Moreover, the JADE platform critically increased the flexibility of the system. In the centralized control model, which was developed using MATLAB, agents were performing optimization sequentially, so it took about 1000 milliseconds. While agents in the JADE model were running simultaneously and simulation time decreased to less than 10 milliseconds for coordination among the same number of power converters, increasing the number of agents in the JADE model did not affect the upper time limit of optimization process. The agent-based strategy achieves coordination in a time frame acceptable for the defined soft real-time application.

7.1. Future Work

Since the real-time agent-based simulator is already developed, it can be used to overcome operation, control, and protection challenges that are caused by deployment of distributed power units in microgrids such as the integration of electric vehicles [123]. The tremendous growth of electric vehicles will affect the secure and steady operation of power systems. In order to make power systems stable, it is essential to analyze the impact of electric vehicles on distribution networks. A research can be defined to analyse the impact of charging stations on distributed power networks. For each charging mode, including normal and fast charge in charging stations, distributed coordinators can be developed using intelligent technology. All of the designed models will be tested on the integrated JADE platform for verification.

References

[1] FP. Sioshansi, “Smart grid: integrating renewable, distributed & efficient energy “, San Diego, California: Elsevier; 2012.

[2] H. Farhangi, “The path of smart grid,” IEEE Power Energy Mag., vol. 8, no. 1, pp. 18–28, Jan. 2010.

[3] F. Eddy, H.B. Gooi, S.X. Chen, “Multi-Agent System for Distributed Management of Microgrids”, IEEE Trans. On Power Systems, pp. 24-34, 2015.

[4] M. Manbachi, M Nasri, B Shahabi, H Farhangi, A Palizban, S Arzanpour, Mehrdad Moallem, Daniel C Lee, “Real-Time Adaptive VVO/CVR Topology Using Multi-Agent System and IEC 61850-Based Communication Protocol”, IEEE Transaction on Sustainable Energy, Vol. 5, No. 2, Apr. 2014, pp. 587-598.

[5] G. Dehnavi, H.L. Ginn III, “Distributed control of orthogonal current components among converters in an autonomous microgrid”, IEEE Energy Conversion Congress and Exposition (ECCE), pp. 974 – 981, 2012.

[6] J D. Bakken, “Smart Grids: Clouds, Communications, Open Source, and Automation”, 1st ed., CRC Press, May 2014.

[7] A. Feliachi, K. Schoder, S. Ganesh, HJ. Lai, “Distributed control agents approach to energy management in electric shipboard power systems”, IEEE Transaction on Energy Conversion, 2006.

[8] IEEE-PES Task Force on Microgrid Control, “Trends in Microgrid Control”, IEEE Treansactions on Smart Grids, Vol. 5, No. 4, July 2014.

[9] H. L. Ginn, F. Ponci, A. Monti, “Multi-agent control of PEBB based power electronic systems” 2011 IEEE Trondheim in PowerTech, pp.19-23 June 2011.

[10] Md Moinul Islam, “Advanced Digital Signal Processing Based Redefined Power Quality Indices, and Their Applications to Wind Power”, 2014.

[11] Md. Islam, H. A. Mohammadpour, A. Ghaderi, C. Brice, and Y. J. Shin, “Time-Frequency Based Instantaneous Power Components for Transient Disturbances According to IEEE Standard 1459-2010,” IEEE Transactions on Power Delivery, vol. 30, no. 3, pp. 1288-1297, Jun. 2015.

[12] A.A. Hamad, H.E. Farag, and E.F. El-Saadany, “A Novel Multiagent Control Scheme for Voltage Regulation in DC Distribution Systems”, IEEE Transaction on Sustainable Energy, Vol. 6, Issue 2, pp.534-545, Apr. 2015.

[13] C. P. Nguyen, A. J. Flueck, “Agent Based Restoration With Distributed Energy Storage Support in Smart Grids”, IEEE Trans on Smart Grid, Vol. 3, No. 2,pp. 1029-1038, June 2012.

[14] Shaw Green, Leon Hurst, Brenda Nangle, Dr. Padraig Cunningham, "Software Agents: A review", 1997.

[15] S.D.J. McArthur, E.M. Davidson, V. M. Catterson ; A.L. Dimeas, N.D. Hatziargyriou, F. Ponci, T. Funabashi, “Multi-Agent Systems for Power Engineering Applications—Part I: Concepts, Approaches, and Technical Challenges”, IEEE Transactions on Power Systems, Vol. 22 , Issue 4, pp. 1743 – 1752, 2007.

[16] K. Krauter, R.Buyya, M.Maheswaran, “A taxonomy and survey of grid resource management systems for distributed computing”, Software: Practice and Experience Volume 32, Issue 2, pp. 135–164, Feb. 2002.

[17] MH. Pishbin, AR. Mohammadi, M. Nasri, “Optimisation of Manufacturing Parameters for an Ni–Ag Fuel Cell Electrode”, Fuel Cells, Vol. 7, Issue. 4, pp. 291-297, 2007.

[18] P. Lilienthal, “How to Classify Microgrids: Setting the Stage for a Distributed Generation Energy Future” , Microgrid News and Insight, Apr. 2013.

[19] S. Attarchi, “An Intelligent System for Energy-Efficient Lighting and Illuminance Control in Buildings”, Simon Fraser University, 2014.

[20] Z. Wang, R. Yang and L. Wang, “Multi-Agent Intelligent Controller Design for Smart and Sustainable Buildings”, 4th Annual IEEE Systems Conference, pp. 277-282, 2010.

[21] F. Bignucoloa, R. Caldona, V. Prandonib, “Radial MV networks voltage regulation with distribution management system coordinated controller”, Electric Power Systems Research, Vol. 78, Issue 4, pp.634–64, April 2008.

[22] H.F. Wang, “Multiagent Co-Ordination for the Secondary Voltage Control in Power System Contingencies,” Proc. Inst. Elect. Eng. Generation, Transm., Distrib., Vol. 148, No. 1, pp. 61–66, Jan. 2001.

[23] F. Bellifemine, A. Poggi, G. Rimassa, “JADE–A FIPA-compliant agent framework”, In Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM), pp. 97–108, 1999.

[24] S.D.J. McArthur, E.M. Davidson, V.M. Catterson, A.L. Dimeas, N.D. Hatziargyriou, F.Ponci and T. Funabashi, “Multi-Agent Systems for Power Engineering Part II,” IEEE Trans. Power Systems, Vol. 22, pp. 1743–1759, Nov. 2007.

[25] A. Saleem, N. Honeth and L. Nordstrom, “A Case Study of Multi- Agent Interoperability in IEC 61850 Environments,” in Proc. Innovative Smart Grid Technologies Conference Europe (ISGT Europe), IEEE PES, Gothenburg, Sweden, Oct. 2010.

[26] H. Fakham, A. Ahmidi, F. Colas and X. Guillaud, “Multi-Agent System for Distributed Voltage Regulation of Wind Generators Connected to Distribution Network,” in Proc. Innovative Smart Grid Technologies Conference Europe (ISGT Europe), IEEE PES, Gothenburg, Sweden, Oct. 2010.

[27] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer, “Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In Proc. SIGMETRICS’2000, Santa Clara, CA, 2000.

[28] J. van Dongen, J.M. van de Mortel-Fronczak and J.E. Rooda , “Heterarchical control of a lithoshop”.

[29] N.A. Duffle, R.S. Piper, B.J. Humphrey, and J.E Hartwick, "Hierarchical and Non-Hierarchical Manufacturing Cell Control with Dynamic Part-Oriented Scheduling," Proc. of the 14th North American Mfg. Research Conf., Minneapolis, May 1986, pp504-507.

[30] S. M. Amin, “For the Good of the Grid”, IEEE Power and Energy Magazine, Vol. 6, Issue. 6, pp. 48-59, Nov. –Dec. 2008.

[31] Visionary Manufacturing Challenges for 2020. Washington, DC: National Academy Press, 1998.

[32] Foundation for Intelligent Physical Agents. Specifications. 1997. Available from http://www.fipa.org.

[33] K. Sycara, A. Pannu, M. Willia mson, and D. Zeng, “Distributed Intelligent Agents”, IEEE Expert, 11(6): 36-46. 1996.

[34] T. Finin and Y. Labrou. “KQML as an agent communication language J.M. In Bradshaw (ed.) Software Agents”, pp. 291-316. Cambridge, MA, 1997.

[35] Foundation for Intelligent Physical Agents. Specifications. 1997. Available from http://www.fipa.org.

[36] M. Shahidehpour, Y. Wang, “Communication and Control in Electric Power Systems", Applications of Parallel and Distributrd Processing. IEEE Press, Willey InterScience, pp. 36-44, 2003.

[37] I.A. Ferguson, “Tourching Machines: Autonomous Agents with Attitudes”, IEEE Computers, Vol. 25, No. 5, pp. 51-55, 1992.

[38] K. Fischer, J.P. Muller and M. Pischel, “A pragmatic BDI Architecture,” Proceeding of ATAL 95, No. LNAI 1037 in Lecture Notes in Artificial Intelligence, Springer Verlag, pp. 203-218, 1995.

[39] D.P. Buse and Q.H. Wu, “IP Network-based Multi-Agent Systems for Industrial Automation, Information Management, Condition Monitoring and Control of Power Systems”, Springer, pp. 21-53, 2006.

[40] W. Shen, “Distributed Manufacturing Scheduling Using Intelligent Agents”, Tutorial, IEEE on Intelligent Systems, 2002.

[41] H. Chen, “An Intelligent Broker Architecture for Context-Aware Systems”, Dissertation at the University of Maryland Baltimore County, Jan. 2003.

[42] M. Klusch, K. Sycara, “Brokering and matchmaking for coordination of agent societies: A survey”, Coordination of Internet Agents, Springer, 2001.

[43] Robert W. Brennan, Douglas H. Norrie, “Metrics for evaluating distributed manufacturing control systems”, Computers in Industry 51 (2003) 225–235, Elsevier.

[44] C. Liu, J. Jung, G.T. Heydt, V. Vittal, “The strategic power infrastructure defense (SPID) system. A conceptual design”, IEEE on Control Systems, pp. 40-52, Aug. 2000.

[45] M. Nasri, H. L Ginn, M. Moallem, “Application of Intelligent Agent Systems for Real-time Coordination of Power Converters (RCPC) in Microgrids”, Energy Conversion Congress and Exposition (ECCE), 2014 IEEE, pp. 3942-9, Sep. 2014.

[46] S. Auchariyamet and S. Sirisumrannukul, “Volt/VAr control in distribution systems by fuzzy multiobjective and particle swarm,” in Proc. 6th Int. Conf. Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON, Pattaya, Thailand, May 2009.

[47] F. Bellifemine, G. Caire, D. Greenwood, “Developing multi-agent systems with JADE”, Textbook, 2007.

[48] R.S. Patil, R.E. Fikes, P.F. Patel-Scheneider, D. McKay, T. Finin, T. Gruber and R. Neches, “The DARPA knowledge sharing effort: progress report.” Third Conf. on Principles of Knowledge Representation and Reasoning, pp 103-114. Cambridge, MA. 1992.

[49] “Object Management Group. 95-11-03: Common Facilities RFP3 Final Draft”, 1995, Available from http://www.omg.org/docs/1995/95-11-03.ps.

[50] “Foundation for Intelligent Physical Agents. Specifications”, 1997. Available from http://www.fipa.org.

[51] K. Sycara, A. Pannu, M. Willia mson and D. Zeng, “Distributed Intelligent Agents”, IEEE Expert, 11(6): 36-46. 1996.

[52] Hybrid multi-agent architecture as a real-time problem-solving model, C. Carrascosa, J. Bajo, V. Julian, J.M. Corchado, V. Botti.

[53] S.D. Parsons, P. Gymtrasiewicz, M.J. Wooldridge,”Game theory and decision theory in agent-based systems”, Springer Science and Business Media,, 2012

[54] S. Bussmann,” An agent-oriented architecture for holonic manufacturing control in: First OpenWorkshop on Intelligent Manufacturing Systems”, Europe, pp. 1-12, 1988.

[55] W. Shen, D.H. Norrie,” Agent-based systems for intelligent manufacturing: a state-of-the-art survey”, Knowledge and Information Systems, Vol. 1, pp. 129–156, 1999.

[56] MM Islam, MR Hossain, RA Dougal, CW Brice, “Analysis of real-world power quality disturbances employing time-frequency distribution”, Clemson University Power Systems Conference (PSC), pp. 1-5, 2016.

[57] R.H. Bordini, “Programming Multi-Agent Systems in AgentSpeak Using Jason”, textbook.

[58] S. Frey, A. Diaconescu, D. Menga, I. Demeure, “A holonic control architecture for a heterogeneous multi-objective smart micro-grid”, IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 21-30, Sep. 2013.

[59] Z. Jiang, “Agent-based control framework for distributed energy resources microgrids”, International Conference on Intelligent Agent Technology (IAT ‘06), pp. 646 – 652, Dec. 2006.

[60] A. Koestler, “The Ghost in the Machine”, New York: The Macmillan Co., 1967.

[61] R.F. Babiceanu, F.F. Chen, “Development and applications of holonic manufacturing systems: a survey”, Journal of Intelligent Manufacturing, Springer, Vol. 17, Issue 1, pp. 111-131, Feb. 2006.

[62] S. Rahman, “Intelligent Distributed Autonomous Power Systems (IDAPS)”, Electrical Power Symposium, 2006.

[63] M. Pipattanasomporn, H. Feroze, and S. Rahman, “Multi-Agent Systems in a Distributed Smart Grid: Design and Implementation”

[64] Brian P. Gerkey, M. J. Mataric, “Sold!: Auction Methods for Multirobot coordination”, IEEE Transactions on Robotics and Automation, vol. 18, No. 5, Oct. 2002.

[65] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content addressable network”, In Proceeding of the 2001 ACM SIGCOMM, pp. 161–172, 2001.

[66] S. Cristina, S. Bianchi, “Sistemas Distribudos”, Presentation Transcript, http://slideplayer.com.br/slide/333488/.

[67] A. Gupta, O.D. Sahin, D. Agrawal, A.E. Abbadi, “Meghdoot: content-based publish/subscribe over P2P networks”, 04 Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, pp. 254-273, 2004.

[68] R. V. Renesse, A. Bozdog, “Willow: DHT, aggregation, and publish/subscribe in one protocol” , Peer-to-Peer Systems III, Springer, 2004.

[69] N.Tajuddin B. Maniymaran, H. A. Jacobsen, “Minimal Broker Overlay Design for Content-Based Publish/Subscribe Systems”, Presentation transcript University of University of Toronto Nov. 2013 (http://slideplayer.com/slide/8724176/).

[70] V. Muthusamy, H.A. Jacobsen, Muthusamy, V., Jacobsen, H.A., “Small–scale peer-to-peer publish/subscribe” in Workshop on Peer-to-Peer Knowledge Management, San Diego, USA, July 2005.

[71] P. Triantafillou, I. Aekaterinidis, “Content-based Publish-Subscribe Over Structured P2P Network”, 1st International Workshop on Discrete Event-Based Systems , 2004.

[72] J. Maenpaa, G. Camarillo, J. Hautakorpi, “A self-tuning distributed hash table (dht) for resource location and discovery (reload)”, Standard, Internet Engineering Task Force (IETF),Sep. 2014.

[73] I. Aekaterinidis, P.Triantafillou, “Pastrystrings: A comprehensive content-based publish/subscribe dht network”, Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS’06), 2006.

[74] D. Tam, R. Azimi, and H. Jacobsen, “Building Content-Based Publish/Subscribe Systems with Distributed Hash Tables”, To appear in (i) the International Workshop on Databases, Information Systems and Peer-to-Peer Computing, September 7-8, 2003, Humboldt University, Berlin, Germany.

[75] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “ A scalable content addressable network”, In Proceedings of the 2001 ACM SIGCOMM, pp. 161–172.

[76] S. Setia, “Distributed Hash Tables (DHTs) Tapestry & Pastry”, Course notes.

[77] M.Nasri, R. Hossain, H.L. Ginn, M. Moallem, “Application of Publish-Subscribe Agent Technology for Real-Time Coordination of Power Converters in DC Shipboard Power Systems”, IEEE Electric Ship Technology Symposium (ESTS) Conference, Alexandria, VA, June 22-24, 2015.

[78] F. Bellifemine, A. Poggi, G. Rimassa, “JADE: a FIPA2000 compliant agent development environment”, Proceedings of the fifth international conference on Autonomous agents, pp. 216-217, 2001.

[79] R. Hossain, HL. Ginn, “Real-time Distributed Coordination of Power Electronic Converters for optimization of DC Shipboard Distribution Systems”, 2015 IEEE Electric Ship Technologies Symposium (ESTS) conference, pp.22-24 , Alexandria, VA, June, 2015.

[80] R. H. Lasseter, P. Paigi, “Microgrid: a conceptual solution”, IEEE 35th Annual Power Electronics Specialists Conference(PESC 04), Vol. 6, pp. 4285 – 4290, 2004.

[81] Shuo Pang, Jay Farrell, Jie Du, and Matthew Barth, “Battery state- of-charge stimation” in Proceedings of the American Control Conference Arlington, VA June 25-27, 2001.

[82] M. Nasri, HL. Ginn, M. Moallem, “Agent-based Real-time Coordination of Power Converters in DC Microgrids”, IEEE Trans. On Smart Grids, Submitted, 2016.

[83] Dan Li, Dougal, R.A., Thirunavukarasu, E., Ouroua, A., “Variable speed operation of turbogenerators to improve part-load efficiency” in the proceedings of Electric Ship Technologies Symposium (ESTS), April 2013 IEEE, Arlington, VA.

[84] H. Fakham, F. Colas, and X. Guillaud, “Real-time simulation of multi-agent system for decentralized voltage regulation in distribution network,” in Proc. IEEE Power and Energy Society General Meeting, Detroit, MI, USA, Jul. 2011.

[85] OPC Server from MatrikonOPC –Modbus and 500 OPC Servers and Products.

[86] SICAM PAS Software (6MD90), Substation Automation System, (http://w3.siemens.com/smartgrid/global/en/products-systems-solutions/substation-automation/substation-automation/pages/sicam-pas.aspx).

[87] RTDS Technologies Inc., (https://www.rtds.com/)

[88] S.P. Meech, “Kansas City Power and Light Company”, The Journal of Political Economy (JSTOR), 1922.

[89] KS Mahajan , “Towards supporting mobility and advanced semantics in Event based systems”, Thesis, 2011.

[90] R. Monson, H.David, A. Chappell, “Java message service”, Publisher: O'Reilly, First Edition January 2001.

[91] J. F. Hübner and R.H. Bordini, “Jason a Java-based interpreter for an extended version of AgentSpeak”, Textbook website (http://jason.sourceforge.net/wp/).

[92] R.H. Bordini, J.F. Hübner, R.Vieira, “Jason and the Golden Fleece of agent-oriented programming”, Chapter Multi-Agent Programming, Volume 15 of the series Multiagent Systems, Artificial Societies and Simulated Organizations, pp. 3-37, 2005.

[93] L. A. Bollinger, M. J. van Blijswijk, G. P. Dijkema, & I. Nikolic, “An Energy Systems Modelling Tool for the Social Simulation Community”, Journal of Artificial Societies and Social Simulation, Vol. 19, Issue 1, 2016.

[94] M. Dickerson, “Agent-based modeling and NetLogo in the introductory computer science curriculum: tutorial presentation”, Journal of Computing Sciences in Colleges, Vol. 30, Issue 5, pp. 174-177, 2015.

[95] A. Banos, C. Lang, & N. Marilleau, “Agent-Based Spatial Simulation with NetLogo”, Vol. 1, 2015.

[96] F. Bellifemine, F. Bergenti, G. Caire, A. Poggi, “JADE—a java agent development framework”, Multi-Agent Programming, Volume 15 of the series Multiagent Systems, Artificial Societies, and Simulated Organizations, pp. 125-147, Springer, 2005.

[97] JAVA Agent DEvelopment Framework (JADE), Open Source Programing Website.

[98] The MathWorks Inc., MATLAB 7, External Interface, 1984-2009.

[99] P. Mendham, T. Clarke, “Macsim: A Simulink Enabled Environment for Multi-Agent ystem Simulation”, 16th IFAC World Congress, Prague,Czech Republic, 2005.

[100] P. Mendham, A. Pomfret, T. Clarke, “Dependable dynamic control using distributed intelligent agents”,5th Congress of the Internatitronal Astronautical Federation, Vancouver, Canada, 2004.

[101] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, “JADE: A software framework for developing multi-agent applications. Lessons learned”, Information and Software Technology, Vol. 50 (2008) 10–2.

[102] “System Design Using Agent Control”.

[103] M. Nasri, R. Hossain, HL. Ginn III, M. Moallem, “Application of Publish-Subscribe Agent Technology for Coordination of Power Converters in DC Shipboard Microgrids”, Accepted, IEEE Trans. On Energy Conversion, 2016.

[104] A. Rowstron1, P. Druschel, “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems”, Springer, LNCS 2218, pp. 329–350, 2001.

[105] M.Nasri**,** R. Hossain, H.L. Ginn, M. Moallem, “Distributed Control of Converters in a DC Microgrid Using Agent Technology”, Power System Conference (PSC), Clemson, SC, March 8-11, 2016.

[106] H. Kopetz, Textbook, "Real Time Systems: Design Principles for Distributed Embedded Applications", (2011).

[107] D. Westermann, M. Kratz, “A Real-Time Development Platform for the Next Generation of Power System Control Functions”, IEEE Trans. On Industrial Electronics, pp.1159 – 1166, Apr. 2010.

[108] M. Torre, “Fundamentals of real-time processing in automation and control”, Consulting Specifying Engineers, Pavel, CA, July 2014.

[109] Ak. Rahideh, M. Gitizadeh and Ab. Rahideh, “Fuzzy Logic in real time voltage/reactive power control in FARS regional electric network,” Electric Power System Research, No. 76, pp. 996-1002, February 2006.

[110] B. Alencar de Souza and A.M.F de Almeida, “Multiobjective Optimization and Fuzzy Logic Applied to Planning of the Volt/Var Problem in Distribution Systems,” IEEE Trans. Power Systems, Vol. 11, pp. 1274–1281, Aug. 2010.

[111] D.H. Spatti, I.N da Silva, W.F. Usida and R.A. Flauzino, “Real-Time Voltage Regulation in Power Distribution System Using Fuzzy Control,” IEEE Trans. Power Delivery, Vol. 25, pp. 1112–1123, April 2010.

[112] E. T. Jauch, “Possible Effects of Smart Grid Functions on LTC Transformers,” IEEE Trans. Industry Applications, Vol. 47, pp. 1013–1021, April 2011.

[113] ANSI Standard C 84.1 – 1995, “Electrical Power Systems and Equipment – Voltage Ratings”.

[114] R. Singh, F. Tuffner, J. Fuller and K. Schneider, “Effects of Distributed Energy Resources on Conservation Voltage Reduction (CVR),” in Proc. Power and Energy Society General Meeting, IEEE, San Diego, CA, USA, July 2011.

[115] M. J. Krok and S. Genc, “A coordinated optimization approach to Volt/VAr control for large power distribution networks,” in Proc. 2011 American Control Conf. O’Farrell Street, San Francisco, CA, USA, Jun. 29–Jul. 1, 2011.

[116] V. Dabic, S. Cheong, J. Peralta and D. Acebedo, “BC Hydro's Experience on Voltage VAR Optimization in Distribution System,” in Proc. Transmission and Distribution Conference and Exposition, 2010 IEEE PES, New Orleans, LA, pp. 1–7, April 2010.

[117] A. Ajaja, “Reinventing Electric Distribution,” IEEE Potentials, Vol. 29, pp. 29-31, Jan.-Feb. 2010.

[118] H.E. Farag, E.F. El-Saadany and R. Seethapathy, “A Two Ways Communication-Based Distributed Control for Voltage Regulation in Smart Distribution Feeders,” IEEE Transaction on Smart Grid, Vol. 3, No. 1, Mar. 2012.

[119] M. Wooldridge, “An Introduction to MultiAgent Systems”, Second Edition, Wiley & Sons, May 2009.

[120] F. Atalla, J. Sayda, and H. Taylor, “A multi agent system for integrated control & asset management of petroleum production facilities,” in Proc. IEEE Int. Conf. Intelligent Control (ISIC 2008), San Antonio, TX, USA,Sep. 2008.

[121] M. Manbachi, “Smart Grid Adaptive Volt-VAR Optimization in Distribution Networks”, Applied Sciences: School of Mechatronic Systems Engineering, Simon Fraser University, Dec. 2015.

[122] M. Nasri, H. Farhangi, A. Palizban, M. Moallem, “Multi-Agent Control System for Real-time Adaptive VVO/CVR in Smart Substation”, IEEE Electrical Power and Energy Conference (EPEC12), London, Ontario, pp.1-7, Oct. 2012.

[123] M. Manbachi, H. Farhangi, A. Palizban, S. Arzanpour, “Real-time communication platform for Smart Grid adaptive Volt-VAR Optimization of distribution networks”, International Journal of Electrical Power & Energy Systems, Elsevier, Vol. 74, pp. 238-251, Jan. 2016.

Appendix A.

Structure of Integrated Systems: RCPC-8, RCPC-32, MG-8, MG-32)

The In this thesis, two types of microgrids are used as the case study. One of them is a middle-sized microgrid including 32 converters, which is used for numerical analysis and selection of the most efficient coordinator control for DC microgrids, called MicroGrid-32 (MG-32) and shown in Figure A.1. The other one is a sample microgrid including eight converters which is used as the case study, modeled in MATLAB, which is called MicroGrid-4 (MG-8) and shown in Figure A.2.

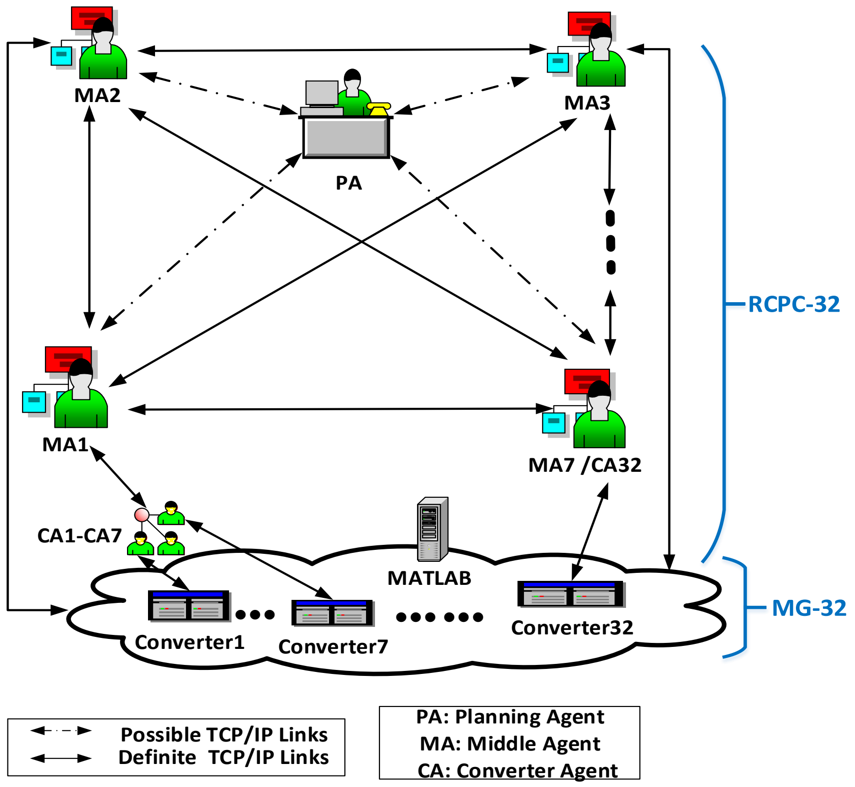


Figure A.1. The Structure of Integrated

System Including RCPC-32 and MG-32

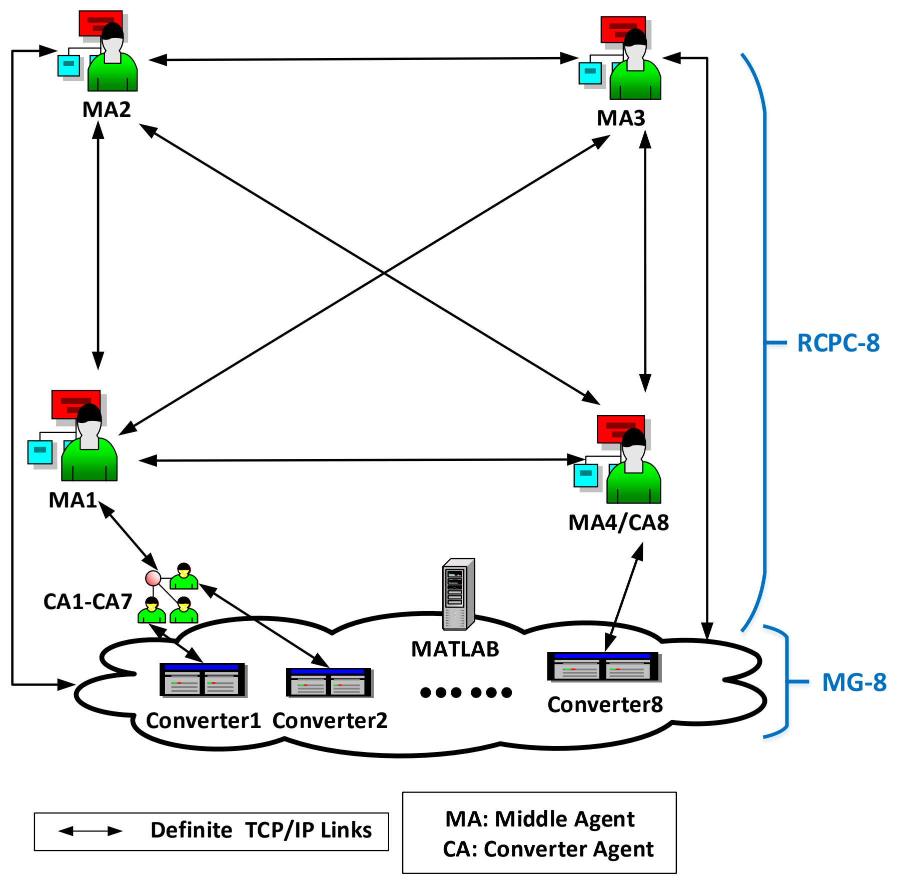


Figure A.2. The Structure of an Integrated System Including RCPC-8 and MG-8

A Real-time Coordinator of Power Converters (RCPC) system is defined as a control model for the sample mocrogrid. To test the applicability of the RCPC optimization method, an MG32 model is used as a case study for numerical analysis which is called RCPC-32. In chapter 6, where a simplified case study including eight converters (MG-8) is used for simulation, its associated agent-based control system called RCPC-8, which is basically a smaller size of RCPC-32. Different control designs are employed in an MG32 system and their efficiency and scalability is compared. In this section, RCPC-32 is designed as a hybrid system to combine the predictability of the centralized and hierarchical control architectures with the agility and robustness against disturbances and a high degree of adaptability of the heterarchical control architectures [1]. Figure A.2 illustrates the application of agent technology in an RCPC-32 system. Each Converter Agent (CA) is connected to one of 32 converters using Ethernet communication protocols (TCP/IP or UDP/IP) located in the lower layer. A group of CAs is assigned to each middle layer agent. The number of Middle Agents (MAs) and the topology of their connection to both CAs at the lower layer and Planning Agent (PA)/MatchMaker Agent (MchAg) at the upper layer is varied for different agent algorithms. There is only one PA that is responsible for saving and mapping system plans.

Appendix B.

The Steps Taken for Implementing the Integrated Control System (RCPC-8)

Implementing the testbed for simulating RCPC-8 takes place in three main steps, detailed in the following sections. The steps are:

1. Developing MAS using JADE in Eclipse. Its shorten pseudo code is explained in Appendix C.
2. Installing MACSim libraries on Eclipse and creating an Agent Task Force class inside the project. Since the original MACSim block should be run through the Windows32 command line, setting this part is complicated and will be explained in detail.
3. Developing the microgrid model on Simulink
4. Running the Simulink model from MAS through MASCimThis is a fairly complex piece of software with many dependencies. Steps 2 through 4 are explained in the following sections.

Installing MACSim on Eclipse

In this research, Eclipse 32 bits v.4.3.0 is opted for developing the case study because of its features. As an MACSim block is only compatible with Windows 32 bits, the path setting of Windows 64 bits is modified to read the system files from a Windows32 folder. In addition, a compatible version of JDK and Eclipse with Windows 32 bits is installed and used for programming. Before deciding to choose Eclipse, a few other main agent development platforms such as Jason, Netlego, and NetBeans are tested and some of their specifications are compared in Table 4.1. As a result, Eclipse is chosen and used for developing a multi-agent control system through the following steps:

1. Add external agent jar files to the Eclipse library. The jar files are: commons-codec-1.3.jar, commons-codec-1.9.jar, http.jar, iiop.jar, jadeTools.jar, macsim.jar, macsimjx.jar
2. Killing the previous processes is running on port 1099 (Jade port) using CurrPorts
3. Creating an Agent Task Force (ATF.txt) script in the JADE project including: project name, number of input and output signals, number of agents, and frequency of running agent model

Running the RCPC-8 System

For running the DC microgrid model, the following steps are required:

1. Running MACSim through Eclipse and selecting the specific Agent Task Force. It opens the JADE remote agent manager.
2. Running macsimjx.jar as a Java application. This task opens the MACSimjx agent platform which calls system pipes.
3. Running the coded agent file through the MACSim platform. Here the file is called Agent1 and its UML is displayed in Figure 4.9.
4. Opening and running the microgrid model in Simulink.

Troubleshooting

1. Displaying error message: “java jade.boot -gui jade is closing down” means the program could not run JADE API from the command line. The error was caused because port number 1099, which is assigned for running JADE programs, is already occupied. To fix the error, an application called “CurrPorts” is used to kill the process running on 1099 and free the port so the IDE jade starts to work properly.
2. Displaying the error message: "Error in S-function 'example/S-Function': S-Function 'MACSim' does not exist"? means the path is not in MATLAB. It can be fixed by adding this folder to the MATLAB path: C:\macsimjx\MyTools\MACSim\client\.

Appendix C.

Pseudo Codes

Initialize agent:

```
    Set up necessary variables (particularly for MIQP optimization algorithm eqs.).
    Determine the number assigned to this agent (provided by AgentCoordinator; it should
    be unique for each agent).
    Set variable: "FirstRun" as true.
    Set initial values for a posteriori estimates,
    by referring to SystemAttributes class.
    Register with DF agent to receive Simulink
    data from AgentCoodinator.
    Start agent behaviour.
```

Agent behavior:

```
    Initialize data structure (named tsd) using
    TimeStepData class (as template for reading and storing incoming data).
    If an "UpdateData" message is received:
        If this is the FirstRun:
            Get address of AgentCoordinator agent.
            Set variable "FirstRun" as false.
        Transfer data from message to tsd data structure.
    Extract z matrix from data structure.
        Calculate local filter estimate:
            Use PredictStage class to calculate
            a priori estimate then use
            this, along with the z matrix to
            determine a posteriori estimate
            using UpdateStage class.
        Amend tsd data structure with a posteriori estimate.

        Return tsd in a message (with ID "ProcessedData") to the sender agent
        (likely to be AgentCoordinator).
         Update the values for previous a posteriori estimates with the current
        ones.
If a "DataAmended" message is received:
(AgentCoordinator has successfully updated its data store with the estimate.)
Reply with "ProcessingComplete" to finish the estimation process for this sample.
    If a "Shutting Down" message is received:
    Remove this agent from the program.
```

Appendix D.

List of Publications Based on This Research

1. **M.Nasri**, MD. R. Hossain, H.L. Ginn, M. Moallem, Application of Publish-Subscribe Agent Technology for Coordination of Power Converters in DC Shipboard Microgrids, IEEE Transactions on Energy Conversion, Accepted for Full Subsmission, 2016.
2. **M.Nasri,** H.L. Ginn, M. Moallem, Agent-based Real-time Coordination of Power Converters in DC Microgrids, IEEE Transaction on Smart Grids, Submitted, 2016.
3. **M.Nasri,** H.L. Ginn, M. Moallem, Distributed Control of Converters in a DC Microgrid Using Agent Technology, Accepted, Power System Conference (PSC), Clemson, SC, March 8-11, 2016.
4. **M.Nasri,** H.L. Ginn, M. Moallem, Application of Publish-Subscribe Agent Technology for Real-Time Coordination of Power Converters in DC Shipboard Power Systems, IEEE Electric Ship Technology Symposium (ESTS) Conference, Alexandria, VA, June 22-24, 2015.
5. **M.Nasri,** H.L. Ginn, M. Moallem, Application of Intelligent Agent Systems for Real-time Coordination of Power Converters (RCPC) in Microgrids, IEEE Energy Conversion Congress and Exposition (ECCE) Conference, Pittsburg, PA, Sep. 2014, pp. 3942-9, 2014.
6. M. Manbachi, **M. Nasri,** B. Shahabi, H. Farhangi, A. Palizban, S. Arzanpour, M. Moallem, D.C. Lee, Real-Time Adaptive VVO/CVR Topology Using Multi-Agent System and IEC 61850-Based Communication Protocol, IEEE Transaction on Sustainable Energy, vol. 5, issue 2, pp. 587-594, Apr. 2014.
7. **M. Nasri**, H. Farhangi, A. Palizban, M. Moallem, Multi-Agent Control System for Real-time Adaptive VVO/CVR in Smart Substation, IEEE Electrical Power and Energy Conference (EPEC12), London, Ontario, pp.1-7, Oct. 2012.