Multi-Agent Based Control and Reconfiguration for Restoration of Distribution Systems with Distributed Generators

Jignesh M. Solanki

MULTI-AGENT BASED CONTROL AND RECONFIGURATION FOR RESTORATION OF DISTRIBUTION SYSTEMS WITH DISTRIBUTED GENERATORS

By

Jignesh M. Solanki

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

September 2006

MULTI-AGENT BASED CONTROL AND RECONFIGURATION FOR RESTORATION OF DISTRIBUTION SYSTEMS WITH DISTRIBUTED GENERATORS

By

Jignesh M. Solanki

APPROVED:

Noel N. Schulz Associate Professor of Electrical and Computer Engineering (Major Professor and Director of Dissertation)	Randolph F. Follett Assistant Professor of Electrical and Computer Engineering (Committee Member)
Herbert L. Ginn III Assistant Professor of Electrical and Computer Engineering (Committee Member)	Stan Grzybowski Professor of Electrical and Computer Engineering (Committee Member)
Nicholas H. Younan Professor of Electrical and Computer Engineering (Graduate Coordinator)	Roger L. King Associate Dean of the College of Engineering

Name: Jignesh M. Solanki

Date of Degree: December 9, 2006

Institution: Mississippi State University

Major Field: Electrical Engineering

Major Professor: Noel N. Schulz

Title of Study: MULTI-AGENT BASED CONTROL AND RECONFIGURATION FOR RESTORATION OF DISTRIBUTION SYSTEMS WITH DISTRIBUTED GENERATORS

Pages in Study: 142

Candidate for Degree of Doctor of Philosophy

Restoration entails the development of a plan consisting of opening or closing of switches, which is called reconfiguration. This dissertation proposes the design of a fast and efficient service restoration with a load shedding method for land-based and ship systems, considering priority of customers and several other system operating constraints. Existing methods, based on centralized restoration schemes that require a powerful central computer, may lead to a single point of failure. This research uses a decentralized scheme based on agents. A group of agents created to realize a specific goal by their interactions is called a Multi-Agent System (MAS). Agents and their behaviors are developed in Java Agent DEvelopment Framework (JADE) and the power system is simulated in the Virtual Test Bed (VTB).

The large-scale introduction of Distributed Generators (DGs) in distribution systems has made it increasingly necessary to develop restoration schemes considering

DG. The separation of utility causes the system to decompose into electrically isolated islands with generation and load imbalance that can have severe consequences. Automated load shedding schemes are essential for systems with DGs, since the disconnection of the utility can lead to instability much faster than an operator intervention can repair. Load shedding may be the only option to maintain the island when conditions are so severe as to require correction by restoration schemes. Few algorithms have been reported for the problem of maintaining the island, even though load shedding has been reported for power systems using under-frequency and under-voltage criteria. This research proposes a new operational strategy for sudden generator-load imbalance due to loss of utility that dynamically calculates the quantity of load to be shed for each island and the quantity of load that can be restored. Results presented in this dissertation are among the first to demonstrate a state-of-the-art MAS for load shedding under islanded conditions and restoration of the shed loads.

The load shedding and restoration schemes developed here have behaviors that can incorporate most of the distribution topologies. Achieving service restoration with DG is complicated but new automated switch technologies and communications make MAS a better scheme than existing schemes.

DEDICATION

I would like to dedicate this dissertation to my wife.

ACKNOWLEDGMENTS

I would like to deeply thank my advisor, Dr. Noel N. Schulz, for her inspiration, enormous support and valuable guidance. She gave me opportunities to interact with other researchers through conferences, meetings and informal get-togethers that enhanced my personality.

I would also like to thank my committee members, Dr. Randolph Follett, Dr. Herb Ginn and Dr. Stan Grzybowski, for their precious suggestions and rich coursework, which made me strong by providing advanced knowledge and software skills. Thanks are also due to Dr. David Cartes at Florida State University, for developing the interface between JADE and VTB.

I would also like to acknowledge the financial support of the Office of Naval Research. Special thanks to all the technical staff and graduate students working on the E-ship project. I would also like to thank Dr. Haibin Wang for his initial work on agents.

I am particularly grateful to the incomparable support of my wife throughout my dissertation. I would like to acknowledge the support and love of my parents and sisters.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

This chapter briefly introduces the research problem by giving a broad overview on restoration, ship systems, distributed generation and agents; it provides a strong base for understanding the rest of this dissertation.

1.1 Introduction to Restoration

Power systems consist of generation, transmission and distribution systems. Transmission lines connect generators and distribution systems. Most of the modeling and analysis has been achieved for power transmission systems and very little attention has been paid to distribution systems. The distribution engineers have fewer software tools to analyze systems and get fast real-time solutions. This research work concentrates not on the normal conditions of the distribution system, but on emergency conditions like outages. Outages and faults are inevitable in distribution power systems. The general practice for alleviating outages or faults is isolation of the faulted part of the power system. In the process of isolation, some unfaulted areas lose power. Restoring power to these out-of-service areas as soon as possible is essential. That process is called restoration. Restoration entails a fast and efficient switch operation scheme that isolates the faulted area and restores the remaining parts of the system.

The system should automatically reconfigure to maintain this, which needs understanding of system dynamics and distributed control. Distribution systems present challenges for robust control and reliable operation. These include

1. Systems are multi-component.
2. They are vulnerable to attacks and local disturbances, which can lead to widespread failure instantaneously.
3. The number of possible interactions increases, as the distribution systems grow complicated. Thus, it exceeds the ability of a single centralized entity to evaluate, monitor, and manage them in real time.
4. Distribution systems are complex for conventional mathematical theories and control methodologies.

Also, with the recent initiatives to enhance the reconfigurability and continuity of supply for distribution systems, much effort is being concentrated on distribution automation. Controlling and directing the power where needed and desired after a fault is an expected outcome of the operation. Restoration is also important problem for the navy that expects to operate ships even in a hostile environment. As a result of battle damage, some loads are cut off and fast restoration of supply to these loads is necessary for system survivability. Restoration in ship systems must consider the different priorities of various loads. A system should shed unnecessary loads and maintain vital loads such as weapons and communication and operate loads needed for survivability. This problem is a combinatorial problem due to the many combinations of switching operations, and it increases with increasing complexity of the system. Thus, the solution for such a problem in real-time is a challenging task.

1.2 Introduction to Ship Systems

Ship systems are finite inertia electric power systems. Generation, transmission, and distribution systems in ship systems are tightly coupled. Shipboard electric power distribution systems use an ungrounded delta-type system configuration. Ungrounded means the system is capacitively coupled to ground through insulating dielectrics, and no direct connection occurs between the reference ground and any system neutral. System reconfiguration for restoration for ship systems requires a proper simulation. Very little information exists on simulating an ungrounded power system, and thereby detecting a fault. The three-phase power is generated and distributed through three-phase cables. The other components of ship systems are load centers, power distribution panels, bus transfer units and transformers. Researchers have proposed an Integrated Power System (IPS) to meet specific ship requirements. Two power buses run longitudinally along the port and starboard side of ship. The ship is divided into electrical zones. Each zone has two Power Conversion Electronic Modules (PCEMs), which distribute power to loads. Transformers convert voltage to a desired level and provide isolation for sensitive loads. Electric ships have large dynamic loads, relative to generator size, due to the limited capacity. Turbine prime movers are faster than utility systems, and the size and proximity of power system components require fast voltage controls [1]. Ship system loads have several priority levels, depending on the importance of the load to the survivability of the ship. These load priorities can change dynamically because after a missile hits, the priority of the operation may change.

This work maintains the radiality of the system. Several advantages occur with maintaining a radial system: lower short circuit currents, simpler switching and the easier design of protective equipment and its coordination.

1.3 Introduction to Distributed Generators

Distributed Generation (DG) was previously described as a backup generator. With increased demands, several small generators are being installed on the distribution side, which have environmental benefits owing to their technology (wind, solar, etc.). DG can help defer some of the costs of the expansion of the grid to accommodate the rising demands; opponents of the DG feel that the savings are very less. Figures 1.1 (a) and (b) show that the DG unit reduces power flow on all equipment between it and the utility [2], thus reducing losses and increasing reliability.

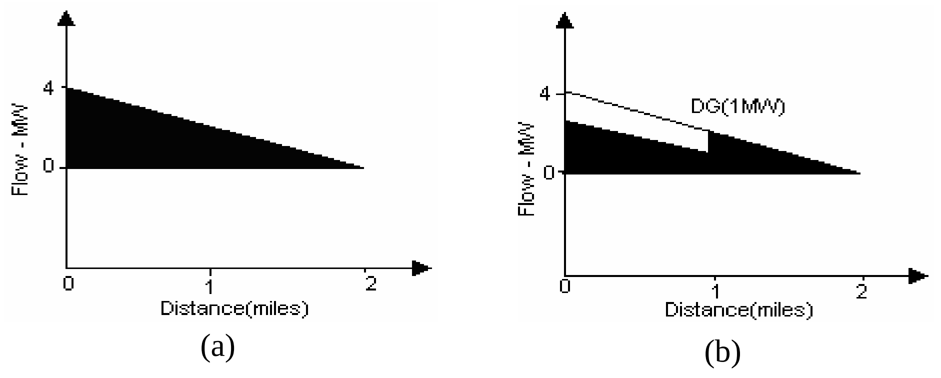


Figure 1.1 Power Flow from Utility (a) with DG (b) without DG [2]

Reference [3] defines DG as “Electric generation facilities connected to an area electric power system (EPS) through a point of common coupling (PCC).” A PCC is a point where local electric power system is connected to an area electric power system. Several other definitions of DG exist [4]:

1. Any qualifying facility under the Public Utility Regulatory Policies Act of 1978
2. Facilities that are automated and modular
3. Facilities that provide other functions in addition to power and capacity
4. Any generator located near a distribution system

This research considers DG as any generation that has a limited capacity and is installed at the customer side or load side.

DG’s contributing to system operations require a command and control interface capable of integrating the data and controlling the resources. The DG could easily be the source of reactive power, eliminating the need for a significant quantity of switching caps, tap-changing transformers, etc., on the main generation. Forcing DG to be in a voltage-control configuration instead of a constant power factor configuration can provide automatic power factor correction. If a large number of DGs have integration of real and reactive power with the main generation, a potential for adequate control theory for large, complex systems (implicitly requiring a significant peer-to-peer communications) guaranteeing robust stability and performance emerges, which does not exist at present.

Fault isolation can result in islanding where a portion of the system that contains both load and DGs is isolated from the utility system while remaining energized.

Intentional islanding attempts to enhance the service by maintaining the continuity of supply to critical/vital loads. For intentional islanding to provide benefits, planning, fault analysis and stability studies must be conducted. For this task, the upstream breaker should open, and DG must be capable of carrying the entire load in the island. A power balance should be prevalent in the island to maintain a suitable voltage and frequency. If a fault occurs in the island, then it should be detected, and islanding should be stopped. After the fault is removed, and before the re-connection of the DG to the main system, the DG should be brought in synchronism with the main source. Several other types of analysis that need to be conducted in the island are stability studies, protection coordination and planning studies. The controls may not be fast enough to act during the process of island formation, hence some kind of coordinated control is needed in the case of several DGs, for a smooth transfer. The terrestrial power system, as well as ship systems, of the future must have autonomous control that is inherently designed to be robustly stable.

IEEE standards are being developed to improve DG operations and connections by setting guidelines. IEEE 1547 outlines standards for interconnection of DG with the grid. The standards provide rules for a sound interconnection in terms of safety, operation limits and performance. IEEE 1547.1 [5] lays out testing procedures to verify that the interconnection complies with the IEEE 1547 standard. The test includes temperature stability, abnormal voltage and frequency, synchronization, unintentional islanding etc. [6] introduces other standards in the series of IEEE 1547.

1.4 Introduction to Agents

Agents are autonomous and can exhibit control over their internal state. An agent is a type of software abstraction. An object is a high-level abstraction that describes methods and attributes of a software component; however, an agent is an extremely high-level software abstraction that efficiently describes a complex software entity. Rather than being defined in terms of methods and attributes, an agent is defined in terms of its behavior and ontology. Multi-Agent Systems (MAS) are distributed and coupled networks of intelligent hardware and software agents that work together to achieve a global goal. Although each agent is intelligent, having only local knowledge restricts its capability, and it alone may not be sufficient to achieve a global goal for a complicated large system. An agent is a computer system capable of autonomous action in some environment.

The agents in this research communicate through a FIPA (Foundation for Intelligent Physical Agents) compliant language [7].

1.5 Objectives of Dissertation

Most of the techniques of restoration need proper problem formulation, and on-the-spot switching is not possible. Restoration requires complete system data at the current system state, and thus a powerful central computing facility to manage real time data. Restoration requires two-way communications and a stand-alone computer able to deal with large amounts of data and information. This requirement can lead to a single-point-of-failure. This dissertation proposes restoration methods based on distributed intelligence techniques. Distributed intelligence deals with concentrating the intelligence on a component level. The multi-agent system (MAS) technology allows the control of a distribution system to be modeled as distributed, autonomous, and adaptive intelligent agents. Each agent may have only a local view of the system, but the team of agents (i.e., multi-agent system) can perform system wide reconfiguration for restoration through autonomous and cooperative actions of the agents. This dissertation focuses on:

- Developing a Multi-Agent System (MAS) for reconfiguration of power systems to restore quickly after isolation of a fault and implementing and testing the developed MAS on land-based distribution systems.

- Extending the MAS developed for restoration of land-based systems to do load shedding and restoration of distribution systems with DG and in case of disconnection of utility maintaining island for continuity of supply to some of the loads that can be supplied by DG. Testing load shedding and restoration on land-based and ship distribution systems. Developing and testing autonomous control schemes for DG output to track load variations and system changes.

1.6 Outline of Dissertation

Chapter 2 describes the problem and surveys the methods found in the literature for the restoration problem. The chapter also gives a brief description of the software packages utilized for this research. Chapter 3 creates the MAS environment in MATLAB and lists the disadvantages of such an implementation and builds necessary load model for a three-phase analysis in Virtual Test Bed (VTB) software. Agents with their behaviors are developed using Java Agent DEvelopment (JADE) toolkit in Chapter 4. The chapter also shows the restoration on several test cases. Chapter 5 describes the achievement of restoration with load shedding in the presence of DG and suggests an autonomous control scheme for DG to track the changes of load. Chapter 6 summarizes the research work and addresses future research problems.

CHAPTER II

LITERATURE REVIEW

This chapter surveys the schemes existing at present for the restoration problem and highlights the essential contributions this research makes by enlisting the differences among existing techniques.

2.1 Restoration

Restoration methodologies can be broadly classified into two categories, centralized and decentralized. In a centralized solution, there is a central control that needs the entire system information. In a decentralized solution, the control is distributed, and actions are based on local information. Several centralized solutions proposed in literature can be broadly categorized into:

- Mathematical Programming
- Meta-Heuristics
- Knowledge-based Programming

Mathematical programming uses the standard traditional optimization techniques like successive linear programming, sequential quadratic programming, lagrangian and other techniques. Butler and Sarma [8] present an optimization-based restoration strategy for all AC ship systems. They used CPLEX to solve the restoration problem and also linearized some of the constraints. Linearizing, however, may lead to suboptimal solution. Khushalani and Schulz [9] presented an optimization-based restoration strategy including distributed generation for an AC-DC ship system. Their method allowed islanding in restoration and kept the problem non-linear. They used LINGO for optimization, which uses the branch and bound technique, as the problem is a mixed integer problem. When traditional optimization techniques are used for restoration in land-based distribution systems, a problem becomes too large to solve in a reasonable time. Also, the restoration problem is a combinatorial optimization problem, which is NP- complete, i.e., finding an optimal solution in reasonable time is not possible. Thus, techniques that fall into Meta-Heuristics are used to solve the above problem of time restriction. These techniques include Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA) and Tabu Search (TS). Of these techniques, GA [10-12], TS and its variations [13-15] and Expert Systems [16-18] have been used extensively for the restoration problem. Also, hybrid techniques [19-21] have been used for this problem. The time constraints for these techniques can be removed by hierarchical optimization [22], which splits the problem into subsystems, optimizing them and coordinating among them, and a hybrid method combining parallel TS and ordinal optimization [23], a probabilistic optimization method that increases the computational speed.

The techniques discussed above do not guarantee an optimal solution with the exception of the time-consuming mathematical programming. These techniques, moreover, are centralized; a small update requires a change in the entire structure. Therefore, this research uses the decentralized technique of multi-agents. The software community first introduced multi-agents as a distributed artificial intelligence scheme. Following this introduction, several papers have commented favorably on the usefulness of agents. Odell [24] uses life as a metaphor for developing agents. He compares agents to living systems and suggests many mechanisms for designing autonomous agents. Jennings [25] notes that a multi-agent has the potential for improving modeling, designing and implementation of computer systems. The author argues that agents with the ability to interact socially can achieve the development of robust and scalable software for dynamic and uncertain environments. Wooldridge and Jennings [26] recognize that agents are being developed and tested for different domains. However, they point that people are becoming dogmatic about agents and state that a non-agent approach should be used since system development engineers can understand them better and that the prototypes of agents are being confused with robust and reliable solutions. They note that with too many agents, emergent functionality may exist that may or may not be good, and with too few agents, the result is like putting all functionalities in a single class in an object-oriented program.

Most of the work in power systems using MAS has been done in the protection area. Yang et al. [27] present five agents, namely Power System Agent (PSA), High Voltage Diagnosis Agent (HVDA), Substation Diagnosis Agent (SDA), Low Voltage Diagnosis Agent (LVDA) and a Global Decision Agent (GDA). They distinguish between these agents based on the voltage level where agents detect the location and type of fault. The authors use Knowledge Query Manipulation Language (KQML) as messaging language. Baran et al. [28] enhanced the EMTP simulation environment for simulating protection and control schemes that use communication between devices. The features of the interface are applied on a pilot protection scheme, which detects a fault based on an overcurrent criterion. Takaya et al. [29] give some example applications of remote monitoring and operating that applies agent technology by introducing a setting agent to change protection and control setting, an analyzing agent to collect the equipment operation information and a patrol agent to gather information about the condition of relay equipment. Wang et al. [30] present back up protection by embedding the agents in protection components to get relay IEDs (intelligent electronic devices). Relay agents monitor their corresponding relays and take corrective actions when failures occur in protection system. Yu and Li [31] define several types of agents, including the protector agent, which isolates a fault and receives the setting of relay obtained from system and valuation agents. Other works [32-34] consider this area of power system protection. Esmin et al. [35] present a prototype MAS with a java-based GUI used to simulate power systems. Their system architecture is composed of a decision package that consists of an Event Identification Agent, a Planning Agent and a Model Agent and a tool package that consists of a Restoration Agent, an Optimal Power Flow Agent, a Load Shedding Agent, a Load Forecasting Agent, an Alarm Processing Agent and a Unit Commitment and Economic Dispatch Agent. Buse et al. [36] point out that many internet-based technologies have emerged, but the problem still remains of handling a large amount of data; thus they find agent-based technology appropriate for power systems, as they are inherently distributed. They have divided MAS into several modules, namely the Global modules and the LAN modules. Sun and Cartes [37] reconfigure the power system network so that the cost of power allocation is minimal. They note that the power flow sometimes settles on a solution where the higher cost sources are used before lower costs are fully utilized and hence they develop an agent-based system to achieve the usage of lower cost sources first. Another work in the field of e-commerce [38] mentions agent-based auctioning as an EPRI project. The authors are building SEPIA (Simulator for Electrical Power Industry Agents), a multiple adaptive agent model of the grid and of industrial organizations that own part of it. Agents in SEPIA are adaptive, and with increasing competition, the simulation can produce optimized pricing structure for a Generating Company (GENCO).

Little research has focused on multi-agents for restoration. A task force for MAS applied to power system was created at the IEEE General Meeting in June 2005. Nagata et al. [39-41] have presented multi-agents for restoration of distribution systems. In [39], the authors classify tie switches into three different categories based on whether they are supplied from the same transformer in the same substation or a different transformer in the same substation, or different substations. The authors have created two types of agents, Load Agents (LAG) and Feeder Agents (FAG), which can act at higher levels. First the LAGs open all the switches in the de-energized area. Then the LAG sends a request for restoration, and FAG stores all the requests and forwards them to other FAGs, which thereby initiate the closing of load switches by LAGs in order to restore. In [40] the authors introduce several types of agents, namely the Independent System Operator Agent (ISOAG), the Local Area Management Agent (LAMA), the Remote Load Facilitator Agent (RLFAG), the Load Facilitator Agent (LFAG), the Generator Facilitator

Agent (GFAG), the Load Agent (LAG) and the Generator Agent (GAG). The restoration is applied to a bulk power system that has three local area networks, including twelve generating units and three remote area networks with twelve loads. The authors report 131 minutes for restoring the de-energized portion of this system. In [41], the authors concentrate on restoration for normal state operation of power systems. They introduce FAGs, Equipment Agents (EAGs) and Switch Box Agents (SBAGs), where the FAG is like a manager, the EAG is a component in the power system, and the SBAG is like a pseudo object consisting of switches and circuit breakers. They tested their program for a single element's maintenance work and multiple elements' maintenance work, on a network consisting of three substations and two high voltage consumers. Wang et al. [42] present a restoration strategy for a test case in one zone of the SPS. They restore power supply to as much non-faulted area as possible, with high priority loads restored first.

The work mentioned above, however, excludes one or all of these:

1. Real time agents' simulation.

2. MAS not totally decentralized.

3. Implementation in FIPA compliant language

This research work focuses on the real-time simulation of a power system in simulation software, with measurements taken by the interface to pass them to the multi-agent network. Thus, real-time restoration is the major focus of this work. A single point of failure can occur if the FAG in [39] fails or the LAMA in [40] fails. This research focuses on a very high level of decentralization, where the agents collaborate to achieve a global goal and continue to do so in the situation of failure of a particular agent. Agents will operate asynchronously where possible, leading to a decrease in communication time and overall speed increase. The scheme will be highly modular; new agents can be added without changes in existing agents.

2.2 Restoration considering Distributed Generator

The introduction of DG in the power system allows the reversal of current in some of the branches. The restoration scheme developed in this research needs no change with introduction of DG unless islanding occurs. Islanding helps improve reliability by maintaining continuity of supply to critical/vital loads. Reference [43] indicates that isolation points must be clearly defined in order to attain the desired reliability improvement. They attempt to reduce the power flow in the interconnection line before separating the utility from the downstream network by disconnecting loads in the downstream network or increasing local generation. Choi et al. [44] propose strategies for restoration in presence of DG where they classify DG in several categories, like black-start DG, non-black start DG, and controllable and non-controllable DG. They suggest that utilities must provide incentives to DG units for providing the communication capability for DG command and control, which is the essential feature for their restoration to succeed. Load shedding is a widely studied topic [45-47]. The load shedding is mostly done under low frequency or low voltage. Reference [48] achieves under-frequency load shedding for a co-generator system connected to unreliable utility. The load shedding is done after the system is islanded in three pre-determined steps in frequency band of 0.7Hz (58.7Hz to 58Hz). The authors also claim, that in the case of an islanded scenario if the load shedding does not occur, blackout or damage to the co-generator may occur after one second. The authors in [49] develop an Intelligent Load Shedding (ILS) scheme for an industrial power system. An ILS scheme consists of a dynamically updating load-shedding table with current and past measured data. The authors also compare the developed ILS with conventional under-frequency and PLC-based load shedding in islanding operation of industrial power system and conclude that their ILS scheme performs optimally and achieves load shedding in less than 100 milliseconds is much better than the conventional load shedding schemes [49]. This research work includes a load shedding scheme under islanded scenario that is:

- fast
- decentralized, that is, based on agents
- independent of pre-determined lookup tables
- able to dynamically control most of the islands created as a result of disconnection of the utility and does not require pre-definition or pre-planning of the island.

Also, the island is then restored from an alternate path if one exists.

2.3 Distributed Generator Control

When restoration occurs, it is assumed that the DG controls are fast enough to act with the changing load. This assumption should not be made, and hence the other focus of this research work will be controlling DG. In reference [50] each DG is equipped with one controller agent and all agents are interconnected to a global agent. During the disturbance agents try to keep the DG in service with regulation of overall system using given set points. The authors simulate three different fault scenarios and show that the system stability is achieved. The authors in [51] investigate the need for a control scheme for DGs due to interaction problem from various kinds of DG when microgrid operates in an islanded mode. Reference [52] describes the effects of DGs on stability of distribution system due to large disturbances. They propose to design the control parameters for the fuel cell by using particle swarm optimization technique. Reference [53] uses frequency control to correct the phase angle deviations caused by load variations. Hiyama et al. [54] present a multi-agent-based control and operation of a distribution system with DG. The authors have achieved load-following control by coordination between a diesel unit and the Energy Capacitor System (ECS). The control is based on the power supplied by the utility due to variations in load. Okuyama et al. [55] present an information-sharing approach between several DGs in an islanded scenario that lead to fast stabilization of the DGs. Hiyama et al. presented a MAS approach for control of DG; however, their MAS architecture does not satisfy FIPA specification and looks like a control unit simply called an agent. This research develops and demonstrates a preliminary work on an autonomous control scheme for DG unit on a test system to load-follow fast. This scheme can then be incorporated into MAS to achieve more speed and capability of acting in any situation in addition to the autonomy.

2.4 Conclusions

This chapter presented the existing methods of restoration. Most of these methods are centralized and require a lot of computation time. A decentralized method of using multiple agents having only local information and coordinating to achieve the required global goal is proposed. Also the methods available for load shedding based on voltage and frequency control are described. The solution proposed in this research work is to do fast load shedding using MAS utilizing mobile agent. A need for autonomous control is described a preliminary work for which will be dealt in Chapter 5. Chapter 3 will focus on the development of MAS for fault detection and restoration and the software packages used for this research. Chapter 4 will use a java platform for developing agents and coordinate with simulation software for measurements.

CHAPTER III

THE RESTORATION PROBLEM AND THE SOLUTION USING A MULTI-AGENT SYSTEM DEVELOPED IN MATLAB

This chapter introduces the restoration problem and mathematically formulates it. A Multi-Agent System (MAS) is developed in MATLAB, but the drawbacks found in such an implementation lead to a different solution.

3.1 Introduction

The components of a distribution system (ship-or land-based) may be connected in the form of a meshed network, but the land-based distribution system is normally operated radially. Four basic strategies for restoration of fault are considered.

1. Fault Detection

On a distribution system, the faults can be of several natures and severity. Some of the common types of faults are single-phase-to-ground faults, phase-to-phase faults, phase-phase-ground and three-phase faults. Several different protection stages can detect the fault severity and location of fault.

2. Fault Isolation

By opening the appropriate circuit breakers/isolators, the faulted section is to be isolated from the rest of the system as quickly as possible.

3. Restoration

After the isolation of the fault, some of the loads in the un-faulted region also get disconnected and are left unsupplied. Service should be restored to these affected loads as quickly as possible by system reconfiguration, which involves changing switch status.

4. Fault Removal

The faulted part needs to be repaired by the crew and the system should be brought to the original configuration.

The concentration of this research work is on the first three stages. The management of disturbances, as well as the effective distributed control of agents, should occur in such a way so that, after a disturbance, parts of the networks will remain operational and even automatically reconfigure themselves. An agent can:

a. Query some of the agents to obtain information
b. Integrate with other agents for decisions
c. Monitor the power system in real-time
d. Perform temporary data logging if essential
e. Offer important messages to other agents

These tasks are achieved through the communication between neighboring agents. In this way, the agent is always informed about its status and its neighboring agent's status. The goal of this research was to design an MAS that fulfills the functional requirements described in this chapter within the discussed constraints.

3.2 Formulation of the Restoration Problem

Restoration involves a set of n binary variables $S = \{S_1 .... S_N\}$ and a set of domains $D = \{D_1 .... D_N\}$, where D_i is a set of values that S can take to satisfy the set of constraints $C = \{C_1 .... C_N\}$. The multi-agent algorithm involves variables V and constraints C distributed among a set of agents $A = \{A_1 .... A_K\}$, where K is the number of autonomous entities. The subset of $D = \{D_1 .... D_{N-K}\}$ and $C = \{C_1 .... C_{N-K}\}$are associated with an Agent A_k. Thus, several tasks are needed to accomplish the restoration strategies as given. The basic functionalities associated with the tasks are:

- Fault Detection

  Data Logging

  Data Analyzing

- Fault Isolation

  Communication with neighboring agents after detecting fault

- Restoration

  Forward and backward tracing communication helps restore the system within constraints, giving priority to vital loads.

Mathematically, the problem is formulated as an objective function satisfying the system constraints. The objective is to maximize the supply of power to as many loads as possible, giving importance to essential loads. This need is satisfied by the weighted sum, $\max \sum_{i=1}^{n} W_i L_i$, where, L is the load current at node i, n is the total number of nodes in the system and W is the weight associated with each load depending on its priority.

The fulfillment of the objective should not lead to violation of the operating limits of the components; this requirement reinforces several constraints.

Limit on Generation

$G_j < G_{\max}$, where j $\in$ S (set of source nodes)

Cable Limits

$F_{ij} < F_{\max}$, where F is the flow between node i and node j.

Radial Configuration

$\sum_{a \in IB} I_a < 1$, where *IB* is the set of inward flow branches incident at a node

The radiality constraint ensures that only one branch feeds the node.

Considering both the constraints of the restoration and the highly distributed system in discussion, a MAS appears to be the best answer to the restoration. The autonomous capability of each agent requires a coordination method among the decisions and actions from a number of agents

3.3 Introduction to Software Packages

This research primarily used three software packages: MATLAB 7.0, Virtual Test Bed (VTB) 1.03.03, and Java Agent DEvelopment Framework (JADE) 3.2. Several other software packages that facilitated communication between VTB and JADE were also used but a description of all of them is beyond the scope of this dissertation.

3.3.1 MATLAB

MATLAB is a matrix laboratory and is a high performance language for technical computing. MATLAB has several application specific toolboxes that are a collection of m-files. MATLAB/SIMULINK is a software package for modeling and simulating linear and non-linear circuits. SIMULINK provides Graphical User Interface (GUI) for building and connecting blocks. One of the blocks is the S-Function block which can be customized and created for the application. Agents were built using the S-Function blocks whose behaviors were governed by M-Files. The details of this implementation are shown in as shown in Section 3.4.2. SimPowerSystems are used to simulate power system blocks in a SIMULINK environment. The load model validation in Section 3.4.1 uses the SimPowerSystems blockset.

3.3.2 VTB

VTB is free software used for simulating large-scale, multi-technical dynamic systems, including electric power, mechanical, thermal and control systems. Apart from performing simulations, VTB also has provisions for building one's own model. VTB's model development guide describes two kinds of models [56]. Interactive models are those that are executed as well as built at run-time, thus facilitating any change needed, but these models are not fast enough. Compiled models are those that need compilation of the code, typically created in C++, thus facilitating a faster run-time, but any change requires recompilation.

The User Defined Device (UDD) is an interactive model that allows creating complex devices using model equations. The compiled model entities can be built using the C++ programming language. There are two basic types of coupling needed for a model entity: natural coupling, which dictates physical conservation laws, and signal coupling, which helps the flow of information. The natural entities are connected with other entities in the system through their ports. Each port is associated with across (voltage) and through (current) variables. The natural entities are classified as linear, nonlinear and linear time-varying entities. The VTB natural models are developed using the Resistive Companion (RC) technique. For the RC method, the equations are expressed as

$$I = GV - B \tag{3.1}$$

Moreover, these equations need discretization; trapezoidal integration is one such method.

$$\begin{aligned} i(t) &= G_1 v(t) - B_1(t-h) \\ 0 &= G_2 v(t) - B_2(t-h) \end{aligned} \tag{3.2}$$

where, i is the vector of through variables, v is the vector of across variables, G is the constant square matrix and B is the vector depending on past history values of through, across and internal state variables. VTB solves these equations [56] using

$$\sum_k A^k i^k(t) = I_{inj} \tag{3.3}$$

where, I_{inj} is the vector of nodal through variable injections, and A^k is a component incidence matrix. Upon substitution of device equations, the system's set of equations becomes a set of nonlinear algebraic equations. These equations are solved using Newton's method. This research developed models as needed. RC modeling needs C++ programming, however, VTB has made several function calls available. A reference to a function call will always contain the default arguments for that specific function. These function calls, with their arguments, are given in [56].

Table 3.1. Functions for VTB Model Development

Functions	Arguments
SetEntityNames()	const std::string &sEntityName,const std::string &sEntityShortName, const std::string &sEntityPrefix
SetEntityComment()	const std::string &sEntityComment
AddLayer( )	None
AddNatureTerminal()	const std::string &sTerminalName,const std::string &sTerminalNature, const std::string &sAcross,const std::string &sThrough,int iLayer = 0,int iRows = 1, int iColumns = 1,bool bOpen = false
AddNatureInternalNode()	const std::string &sNodeName,const std::string &sNodeNature, const std::string &sAcross,const std::string &sThrough,int iLayer = 0,int iRows = 1, iColumns = 1,bool bOpen = false
AddParameter()	const std::string &sParameterName,const std::string &sParameterValue, const std::string &sParameterUnits,const std::string &sParameterComment,int iType, int iRows = 1, int iColumns = 1
AddViewable()	const std::string &sViewableName,const std::string &sViewableValue, const std::string &sViewableUnits,const std::string &sViewableComment,int iType, int iRows = 1, int iColumns = 1,int iPlotOption = 0

Table 3.2. Constructors for VTB Model Development

Constructor	Arguments
UpdateParameters()	None
GetVectorIcon()	None
UpdateModel()	None
ExtractViewables()	None

Table 3.1 gives the specifically needed function calls. The constructor describes the type of the model and is called when that entity is dropped into the schematic. Table 3.2 lists constructors. An entity’s internal variables must be initialized when the simulation begins.

3.3.3 JADE

The Java Agent DEvelopment Environment (JADE) has been implemented in Java language. JADE is a middleware for the development and run-time execution of peer-to-peer applications that use agents. Agent communications in JADE are via message passing, which gives the flexibility for the agent to perform other tasks and not just wait for the message; the messages can be easily retrieved from a queue of messages. Also, it allows the agent to select messages according to its own priority. Communication among agents in JADE is carried out according to FIPA specifications. An instance of the JADE is created when the execution begins, and it is called a container because it contains agents. Several containers make a platform. An agent is executed as a single thread; however, the multithreading nature of JAVA language helps in case an agent needs to execute several tasks in parallel.

JADE implements an agent. The several types of agents [57] are broadly classified as:

Software Agents: Software program that can perform specific tasks for a user and possess a degree of intelligence that permits it to perform parts of its tasks autonomously.

Mobile Agents: Mobile agents are software agents capable of moving from one machine to another automatically.

Reactive Agents: Agents that only react to external stimuli and do not communicate directly.

Interface Agents: Improves interaction between the user and the computer system, e.g., assisting or accessing information or doing learning.

Middle Agents or Intermediate Agents: They are internal agents, which help, in intermediate services. They are Brokers, Controllers or Facilitators

There are several desirable characteristics [57] of an agent as listed below

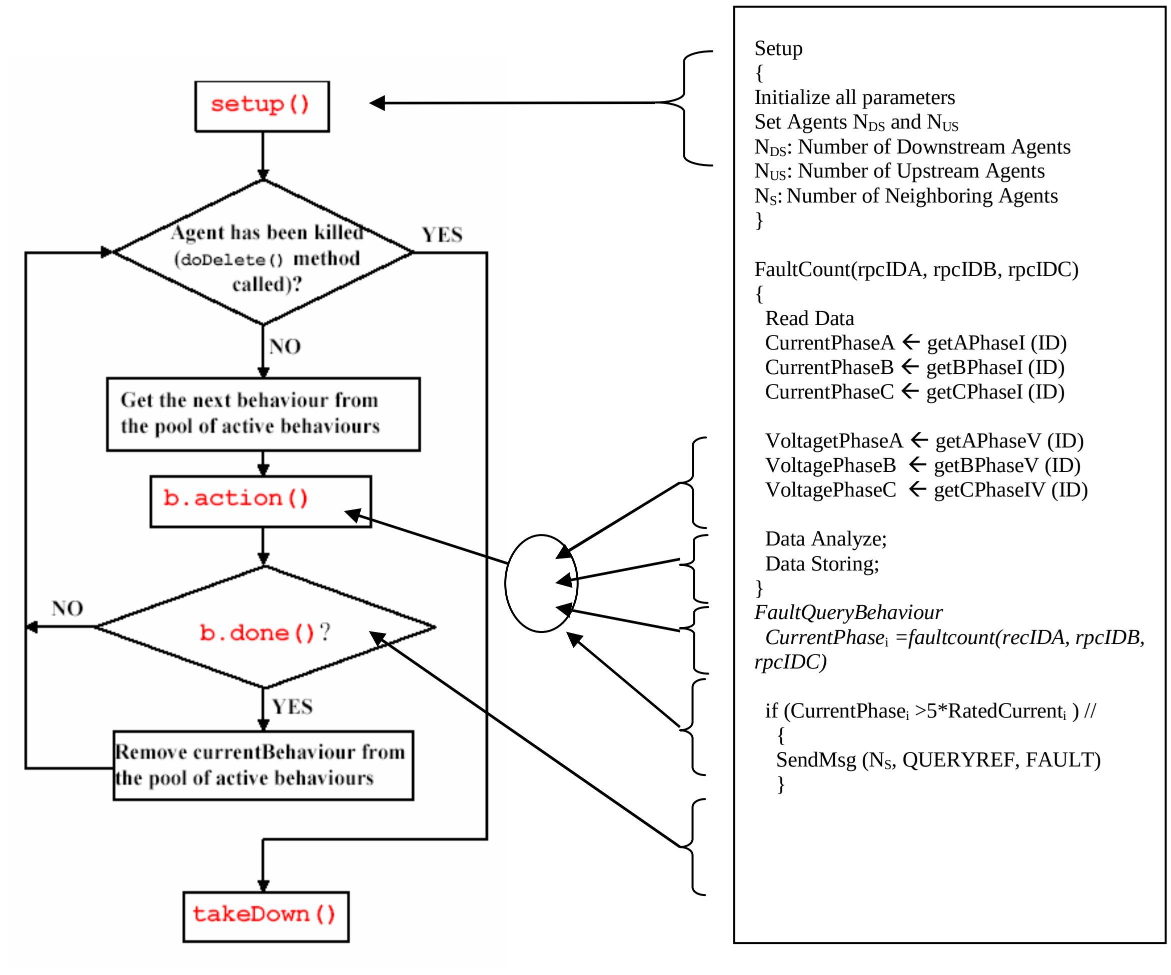
Network-centric	– distributed and self-organizing
Communicative	– interacts with other agent
Semiautonomous	– controlled by humans or other agents
Reactive	– reacts timely and appropriately
Deliberative	– capable of sufficient deliberation
Collaborative	– interacts with human and other agents
Pro-active	– takes initiative, generates goals, acts rationally
Predictive	– makes prediction about future
Adaptive	– accommodates changing task environment
Mobile	– moves around network

Agents execute through behaviors, which are private to each agent and communicate through FIPA-specified Agent Communication Language (ACL). JADE converts the local name of the agent into a globally unique AID (Agent ID). Agents communicate by asynchronous message passing, whose structure follows FIPA specifications. Some standard predefined so-called “types” of messages occur, such as PROPOSE, QUERY_REF, INFORM and REQUEST. In order to distinguish messages, in case of confusion between two behaviors attempting to receive the same message, templates are defined. ACL messages passed between agents are characterized by:

- The Type of Message

- Content of Message
- Intended Receivers
- Conversation ID

Through out the execution the agent as a software entity goes through several stages as shown in Figure 3.1 (a) and programming highlights are as shown in Figure 3.1 (b).


setup()
Agent has been killed (doDelete() method called)?
YES
NO
Get the next behaviour from the pool of active behaviours
b.action()
NO
b.done()?
YES
Remove currentBehaviour from the pool of active behaviours
takeDown()

```
Setup
{
Initialize all parameters
Set Agents N_DS and N_US
N_DS: Number of Downstream Agents
N_US: Number of Upstream Agents
N_S: Number of Neighboring Agents
}

FaultCount(rpcIDA, rpcIDB, rpcIDC)
{
 Read Data
 CurrentPhaseA ← getAPhaseI (ID)
 CurrentPhaseB ← getBPhaseI (ID)
 CurrentPhaseC ← getCPhaseI (ID)

 VoltagetPhaseA ← getAPhaseV (ID)
 VoltagePhaseB  ← getBPhaseV (ID)
 VoltagePhaseC  ← getCPhaseIV (ID)

 Data Analyze;
 Data Storing;
}
FaultQueryBehaviour
 CurrentPhase_i =faultcount(recIDA, rpcIDB,
rpcIDC)

 if (CurrentPhase_i >5*RatedCurrent_i ) //
   {
   SendMsg (N_S, QUERYREF, FAULT)
   }
```

(a) (b)

Figure 3.1 (a) Agent Execution [58] (b) Programming Highlights

The agent's initialization is carried out in setup and execution of behaviors occurs in action. After the action, if the agent's behavior is finished that behavior is removed from the pool of behaviors, and rest of the behaviors continue execution as they are. The complete agent can be killed in the takedown stage.

3.3.4 Communication of VTB with JADE

Figure 3.2 shows the communication structure of JADE and VTB as modified from [58]. It also shows platform JADE, which contains several containers where agents live. The platform is built in JAVA, and JAVA communicates with Microsoft Internet Information Services (IIS), which in turn communicates with VTB through the Remote Procedural Call (RPC) model. This model acts as a Windows RPC server and uses function calls GetSignal and SetSignal from the IIS web server.

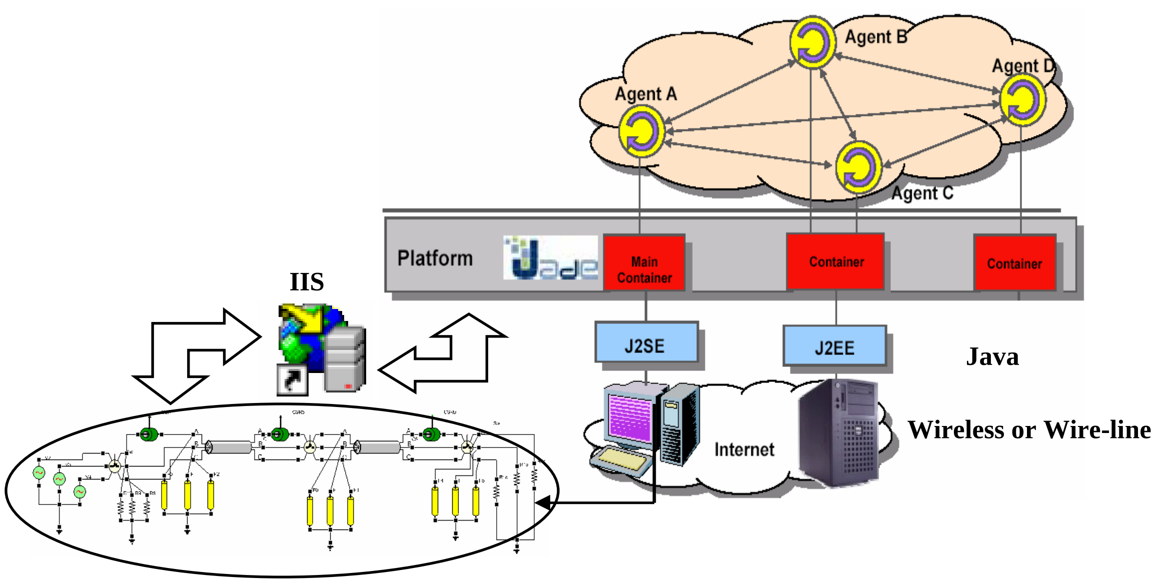


Figure 3.2 VTB-JADE Communication

These two functions are posted on the web using Simple Object Access Protocol (SOAP). Reference [59] describes in detail the communication scheme between RPC and IIS. However, here the description is specific to the research needs. The GetSignal function, also provided by IIS, returns the value of a signal with a specific ID in the VTB schematic, and the SetSignal function provided by IIS sets the value of a signal with specified ID in VTB. The function descriptions when called from the software are doubleGetSignal (short id) and doubleSetSignal (short id, double value). The IIS web server should be started first for this communication. Browser settings need to be changed and the Visual Basic file, which serves the above methods, needs to be compiled and built. Visual Studio .Net, Java 2 Runtime Environment, Java2sdk, Java Web Start and Jade 3.2 were installed for the agent environment.

3.4 Restoration using MAS developed in MATLAB

This section describes the development of the agents in MATLAB and the simulation of the power system in VTB. The power system simulation in VTB needs a three-phase PQ load model, and this section describes the development of such a model. To achieve restoration, the agents are developed in MATLAB, then a fault detection strategy with MAS is developed. MATLAB-VTB co-simulation is achieved, and finally a test case system illustrates the results of such a restoration scheme.

3.4.1 Three-Phase Balanced/Unbalanced PQ Load Model

VTB provided the environment for the development of a three-phase PQ load model using RC technique, for the simulation of three-phase power system. For RC modeling, C++ language was used. Functions provided for model development were used to develop the PQ load model considering resistive and inductive part of the load; a capacitive part was not considered. First, an icon for the model was created in VTB Icon Editor, as shown in Figure 3.3.

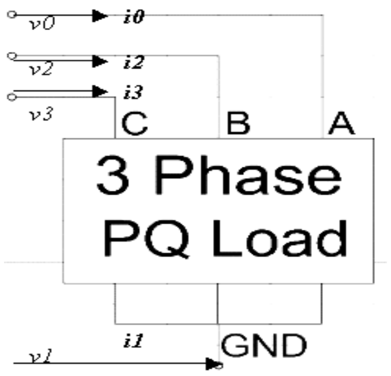


Figure 3.3 Icon of Three-Phase PQ Load

The model equations are:

$$R_a = \frac{V^2 P_a}{P_a^2 + Q_a^2} ; X_a = \frac{V^2 Q_a}{P_a^2 + Q_a^2} ; \; L_a = \frac{X_a}{2 * \pi * 60} \tag{3.4}$$

Similarly, R_b, L_b and R_c, L_c can be calculated. Assuming all phases are uncoupled, they are,

$$v_0 - v_1 = R_a * i_0 + L_a * \frac{di_0}{dt} \tag{3.5}$$

$$v_2 - v_1 = R_b * i_2 + L_b * \frac{di_2}{dt} \tag{3.6}$$

$$v_3 - v_1 = R_c * i_3 + L_c * \frac{di_3}{dt} \tag{3.7}$$

$$i_1 = -i_0 - i_2 - i_3 \tag{3.8}$$

Applying the trapezoidal method on Equations (3.6) to (3.8) produces

$$i_0(t)=\frac{h/2}{(\frac{R_a h}{2}+L_a)}v_0(t)-\frac{h/2}{(\frac{R_a h}{2}+L_a)}v_1(t)-b_0 \tag{3.9}$$

$$where, b_0=-\frac{h/2}{(\frac{R_a h}{2}+L_a)}v_0(t-h)+\frac{h/2}{(\frac{R_a h}{2}+L_a)}v_1(t-h)+\frac{(\frac{R_a h}{2}-L_a)}{(\frac{R_a h}{2}+L_a)}i_0(t-h)$$

$$i_2(t)=\frac{h/2}{(\frac{R_b h}{2}+L_b)}v_2(t)-\frac{h/2}{(\frac{R_b h}{2}+L_b)}v_1(t)-b_2 \tag{3.10}$$

$$b_2=-\frac{h/2}{(\frac{R_b h}{2}+L_b)}v_2(t-h)+\frac{h/2}{(\frac{R_b h}{2}+L_b)}v_1(t-h)+\frac{(\frac{R_b h}{2}-L_b)}{(\frac{R_b h}{2}+L_b)}i_2(t-h) \tag{3.11}$$

$$i_3(t)=\frac{h/2}{(\frac{R_c h}{2}+L_c)}v_3(t)-\frac{h/2}{(\frac{R_c h}{2}+L_c)}v_1(t)-b_3 \tag{3.12}$$

$$where, b_3=-\frac{h/2}{(\frac{R_c h}{2}+L_c)}v_3(t-h)+\frac{h/2}{(\frac{R_c h}{2}+L_c)}v_1(t-h)+\frac{(\frac{R_c h}{2}-L_c)}{(\frac{R_c h}{2}+L_c)}i_3(t-h)$$

$$i_1(t)=-\frac{h/2}{(\frac{R_a h}{2}+L_a)}v_0(t)+\left[\frac{h/2}{(\frac{R_a h}{2}+L_a)}+\frac{h/2}{(\frac{R_b h}{2}+L_b)}+\frac{h/2}{(\frac{R_c h}{2}+L_c)}\right]*v_1(t)-\frac{h/2}{(\frac{R_b h}{2}+L_b)}v_2(t)-\frac{h/2}{(\frac{R_c h}{2}+L_c)}v_3(t)-b_1 \tag{3.13}$$

where, $b_1=-b_0-b_2-b_3$

The Resistive Companion Model Equation using Equations (3.9) to (3.13) is

$$i=Gv-b$$

where, $i=\begin{bmatrix} i_0(t) \\ i_1(t) \\ i_2(t) \\ i_3(t) \end{bmatrix}$; $v=\begin{bmatrix} v_0(t) \\ v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix}$; $b=\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$ and

$$G=\begin{bmatrix} \dfrac{h/2}{(\frac{R_a h}{2}+L_a)} & -\dfrac{h/2}{(\frac{R_a h}{2}+L_a)} & 0 & 0 \\ -\dfrac{h/2}{(\frac{R_a h}{2}+L_a)} & \left[\dfrac{h/2}{(\frac{R_a h}{2}+L_a)}+\dfrac{h/2}{(\frac{R_b h}{2}+L_b)}+\dfrac{h/2}{(\frac{R_c h}{2}+L_c)}\right] & -\dfrac{h/2}{(\frac{R_b h}{2}+L_b)} & -\dfrac{h/2}{(\frac{R_c h}{2}+L_c)} \\ 0 & -\dfrac{h/2}{(\frac{R_b h}{2}+L_b)} & \dfrac{h/2}{(\frac{R_b h}{2}+L_b)} & 0 \\ 0 & -\dfrac{h/2}{(\frac{R_c h}{2}+L_c)} & 0 & \dfrac{h/2}{(\frac{R_c h}{2}+L_c)} \end{bmatrix}$$

(3.14)

Comparing the simulation results with MATLAB/Simulink provides a way to validate the developed model. For this validation, three single-phase PQ loads were connected in MATLAB, as shown in Figure 3.4; Figure 3.5 shows the results. Figure 3.6 shows the VTB schematic for validating the developed PQ load model, and Figure 3.7

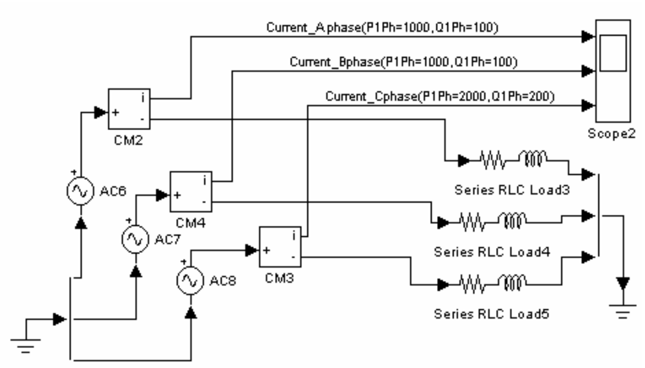

Current_Aphase(P1Ph=1000,Q1Ph=100)
Current_Bphase(P1Ph=1000,Q1Ph=100)
Current_Cphase(P1Ph=2000,Q1Ph=200)
Scope2
CM2
AC6
CM4
AC7
CM3
AC8
Series RLC Load3
Series RLC Load4
Series RLC Load5

Figure 3.4 Three-Phase Unbalanced Load in MATLAB/Simulink

shows the results. Comparing Figures 3.5 and 3.7 validates the developed model as they give the same results.

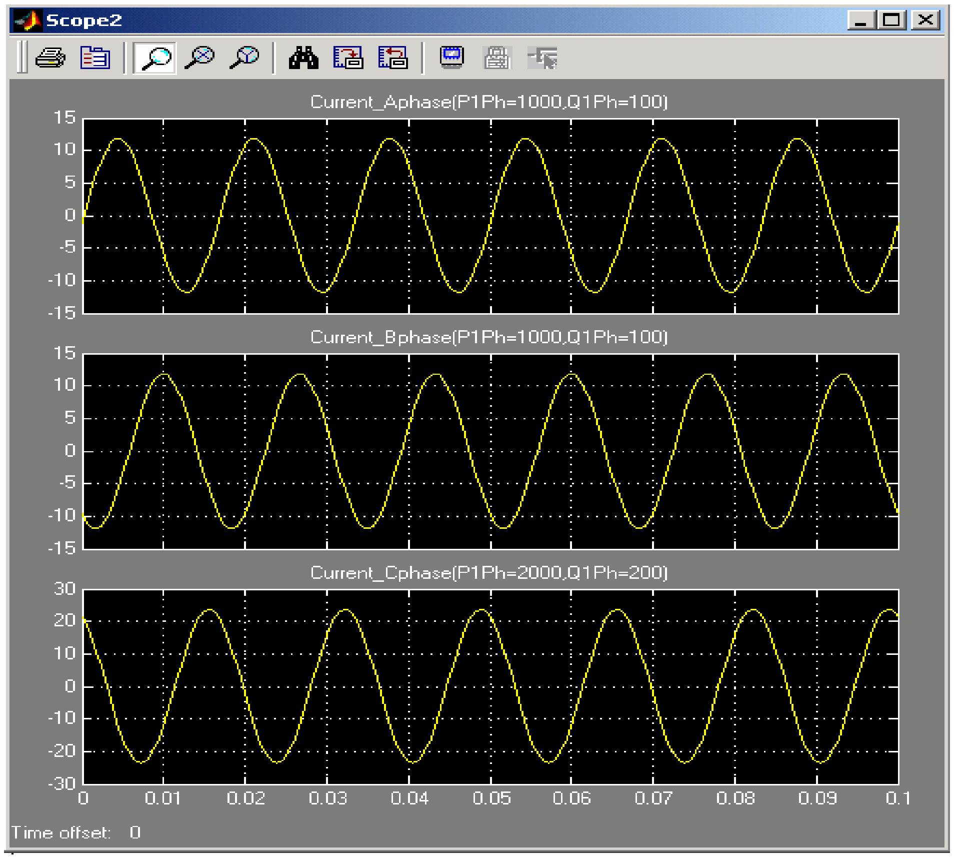


Figure 3.5 Three-Phase Currents of unbalanced PQ Load in MATLAB

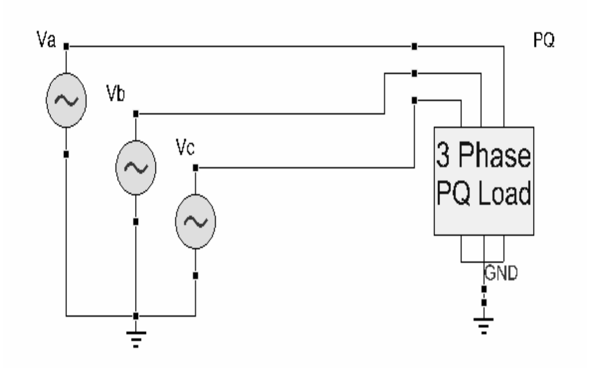


Figure 3.6 VTB schematic of Three-Phase PQ Load

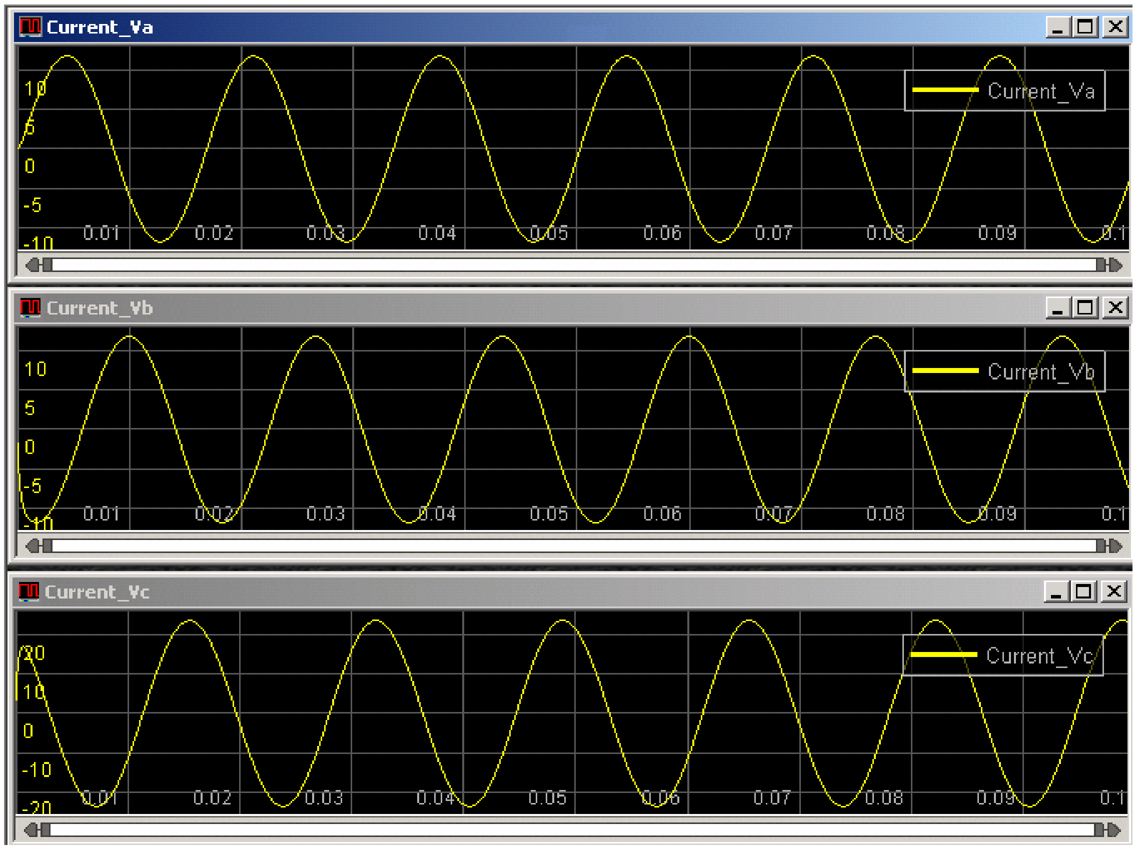


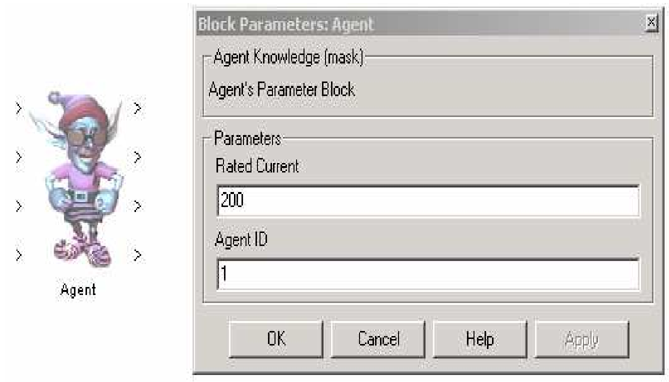
Figure 3.7 Three-Phase Currents of PQ Load

3.4.2 Development of Agents

Agents are implemented using the MATLAB function block in Simulink. Three types of agents are in the proposed MAS: Switch Agents (SAs), Substation Breaker Agents (SBAs) and Tie Breaker Agents (TBAs).

1. Substation Breaker Agents (SBA)

Each SBA can sense the voltage and current magnitude downstream. It prompts for its ID and rated current that it can carry, as shown in Figure 3.8. Table 3.3 shows the inputs and outputs of a SBA. Its initial status is closed.



Block Parameters: Agent

Agent Knowledge (mask)

Agent's Parameter Block

Parameters

Rated Current

200

Agent ID

1

Figure 3.8 Agent Block Parameters

2. Switch Agents (SAs)

Each switch agent can sense the current at the downstream end of the switch, as shown in Figure 3.9. The SA has the information about rated current and the connectivity information about its neighboring two agents. Table 3.3 shows the inputs and outputs of a SA. Its initial status is closed.

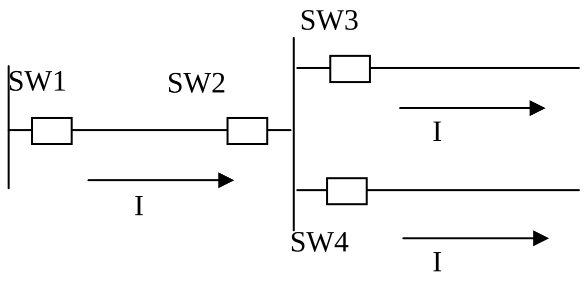


Figure3.9 Illustration of SAG

3. Tie Breaker Agents (TBA)

Every TBA can sense the voltage and current magnitude downstream. It prompts for the ID of neighboring agent and rated current that it can carry. Table 3.3 shows the inputs and outputs of SBA agent.

Table 3.3. Agent Input and Output Signals

Type of Agent	Inputs	Outputs
SBA	present_current, Fdetect_neighbor, Replyreceive, receivemsg	Fdetect, reply_msg, sendmsg, sendself,
SAG	present_current, Fdetect_neighbor, Replyreceive, receivemsg, replyrecieve4	Fdetect, reply_msg, sendmsg, sendself,
TBA	present_current, Fdetect_neighbor, sendself_1, sendself_2, sendself_3	Fdetect, reply_msg, sendself,

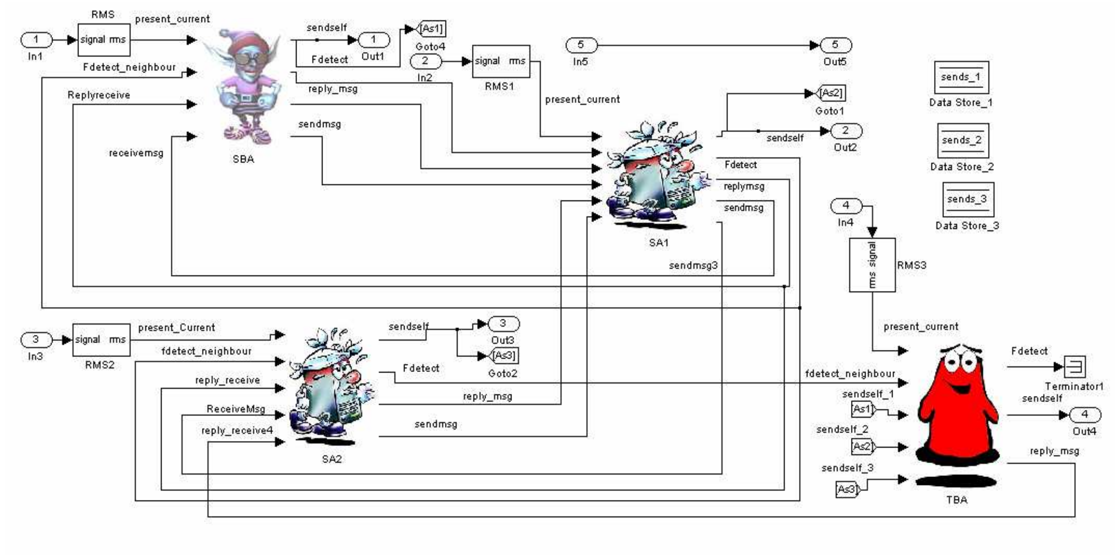


Figure3.10. Agent Architecture

The complete agent architecture is as shown in Figure 3.10. In1 to In5 are the measurement signals, which come from VTB to MATLAB/Simulink, and Out1 to Out5, are the control signals going from MATLAB/Simulink to VTB. This MAS detects the fault, isolates the fault, and restores as many loads as possible.

At the same time, some constraints should be met:

a. No components in the system are overloaded.

b. Radial configuration must be kept.

3.4.3 Fault Detection Strategy

During the reconfiguration, switches should not close on a fault. So the switches that bound the faulted area should be opened and not allowed to close. If a switch agent senses the fault current, it will communicate with all its downstream switch agents. If any of the downstream switch agents also senses the fault current, then the fault is not in the area between these switches, so none of these switches will be locked out. If none of the downstream switch agents senses the fault current, then the fault is just between these switches; so all these switches are locked out. Assume there is only one three-phase fault, Fault 1, as shown in Figure 3.11. SBA1 senses the fault current and communicates with SA2; SA2 also senses that fault current, and so the fault is not between SBA1 and SA2. SA2 senses the fault current and communicates with SA3 but SA3 does not sense the fault, and so the fault is between SA2 and SA3. These two switches are opened and locked out. This strategy works well for the over-current protection scheme.

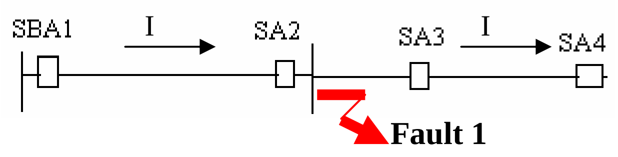


Figure 3.11 Fault Detection

3.4.4 MATLAB-VTB Co-Simulation

In order to test the agent restoration scheme, a power system is simulated in VTB. The MAS needs to monitor the voltages and currents in the power system and control the opening and closing of the switches.

MATLAB is the platform for building the intelligent MAS that makes decisions on the system restoration. Real-time communication is necessary between the simulated power system in VTB and the MAS in MATLAB. VTB allows co-simulation with MATLAB/Simulink through dedicated wrapper entities, as shown in Figure 3.12. Figure 3.13 is a VTB schematic of power system. The simulation in this research work is an interactive signal VTB simulation.

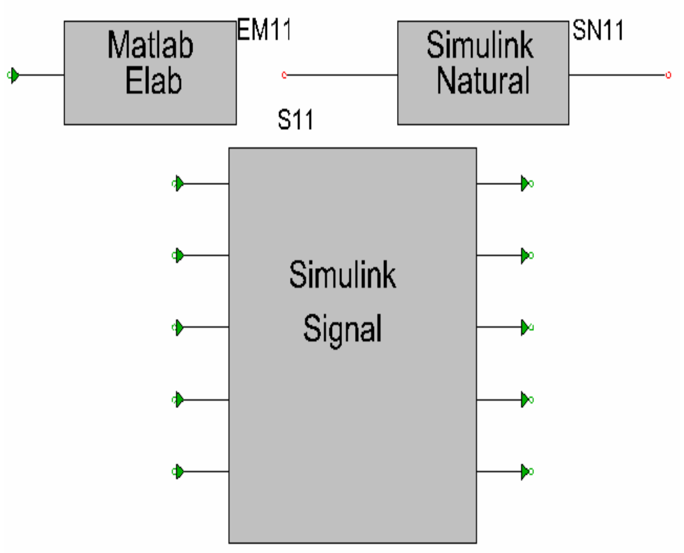


Figure 3.12 Available Wrappers for MATLAB in VTB

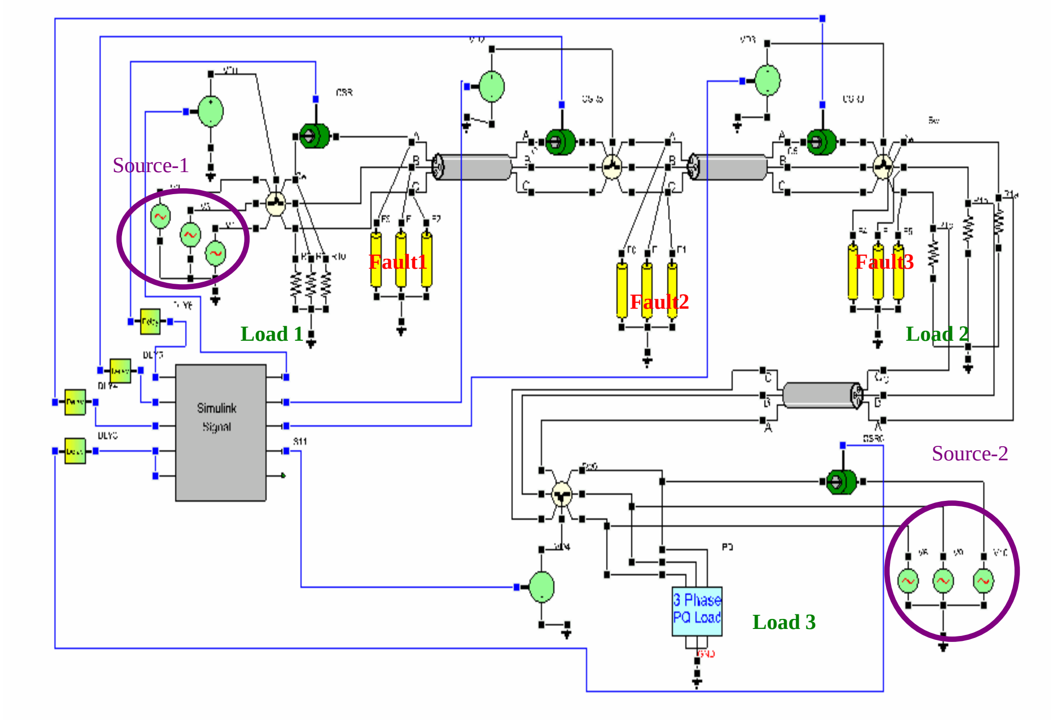


Figure 3.13 Test System I

The agents are created in Simulink and imported to the VTB using the VTB wrapper entity “Simulink Signal”.

Table 3.4 gives the necessary input and output signals needed for the co-simulation of the system in Figure 3.13. During the simulation, MATLAB starts automatically and runs concurrently with the VTB software.

Table 3.4. MATLAB-VTB Co-Simulation Signals

Substation	VTB Signals			MATLAB Signals		
Substation A	Phase A Current from Sensor 1	Phase A Current from Sensor 2	Phase A Current from Sensor 3	Control Signal SW1	Control Signal SW2	Control Signal SW3
Substation B	Phase A Current from Sensor 1			Control Signal SW1		

3.4.5 Results

The small power system shown in Figure 3.13 was tested. Table 3.5 gives the parameters of the system. This test system is a radial three-phase distribution system with a substation transformer being ignored for simplicity. Sources 1 and 2 are the ideal voltage sources. The capacity of all the cables is also set to a very large value. It has resistive and unbalanced PQ loads. Three faults are simulated in the system one by one.

1. Fault 1

SBA sees the fault, communicates with SAG and isolates according to the fault detection strategy described in Section 3.4.3. Switches 1 and 2 open and lockout, thereby isolating the fault. Loads 1 and 2 lose power. TBA gets “open” from SBA and SA1 and it sends “close” to itself. Since TBA detects that its present current is less than 1.5 times its rated capacity it remains closed and Load 2 is restored. Load 1 is not restored because Switch 2 has been open and locked out. Figure 3.14 shows the control signals from agents. Figures 3.15 and 3.16 show detailed results for this fault.

Table 3.5. Test System I Parameters

Elements	**Sub-Station A**	**Sub-Station B**
3-Phase AC Voltage Source	100 Volt Peak	100 Volt Peak
Load 1	2 Ohm/Phase	Pa = Pb = 500 Watt, Qa = Qb = 100 Var, Pc = 1000 Watt and Qc = 500 Var;
Load 2	10 Ohm/Phase	-
Cables	100 meter with Default Parameters in VTB	200 meter with Default Parameters in VTB

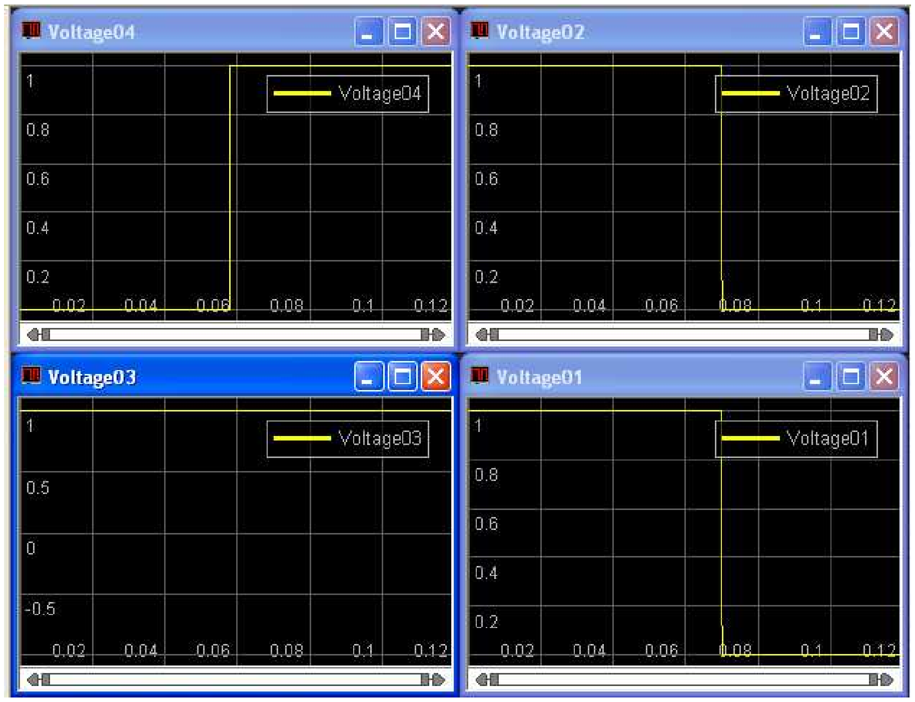


Figure 3.14 Control Signal from TBA, SA2, SA1 and SBA for Fault 1 (starting with upper left corner and reading column wise)

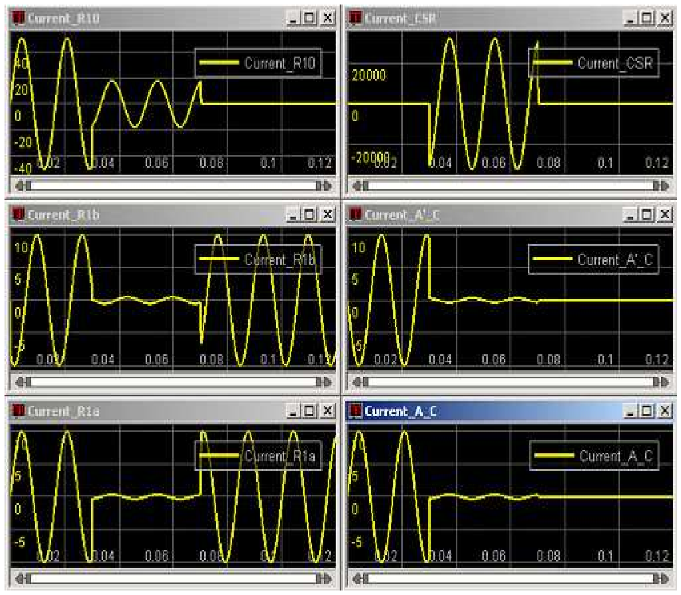


Figure 3.15 Currents: Load 1, Load 2 b and a phase, Sensor and Cable1, Cable 2 (starting with upper left corner and reading columnwise)

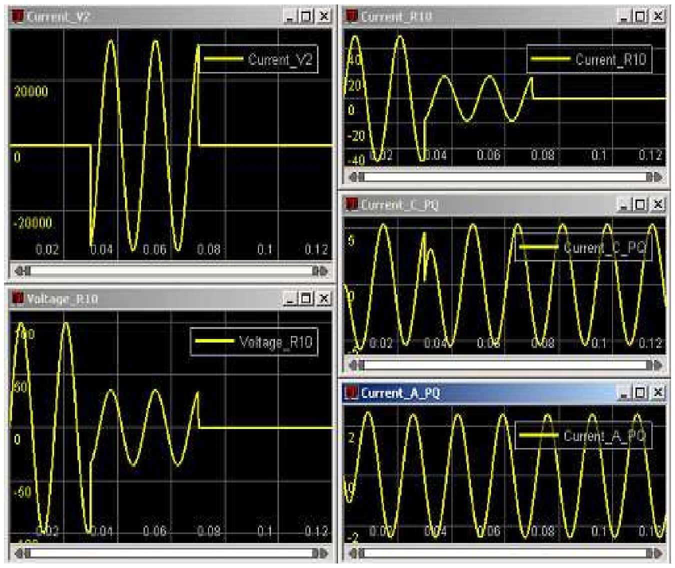


Figure 3.16 Substation A current, Load 1 voltage, Load 1 current, Substation B Voltage, Load 3 Current a and c phases (starting with upper left corner and reading columnwise)

2. Fault 2

In this case, Switches 2 and 3 are opened and locked out, thereby isolating the fault. Load 2 loses power. TBA gets “open” from SA1 and SA2 and it sends “close” to itself. Since the tiebreaker detects that the present current is less than 1.5 times its rated capacity it remains closed, and Load 2 is successfully restored.

3. Fault 3

In this case, Switch 3 opens and locks out thereby isolating the fault. Load 2 loses power. TBA gets “open” from SA2 and sends “close” to itself. However, as soon as it closes, it detects that it is closing into a fault because the present current is greater than 1.5 times its rated capacity, and it opens up.

3.5 Drawbacks for MAS Developed in MATLAB for Large-Scale Systems

MATLAB is a single-threaded programming and cannot execute two or more programs in parallel. In SIMULINK, run time changes are not allowed and storage of data more than once is not possible. Due to its single-threaded nature, peer-to-peer message transfer is not possible. Also, it does not have message storing, polling and queuing architecture. An agent must wait for its action up to next simulation time step, and it does not get updated data at current simulation time step. Distributed control across machines is not available. Run time changes in agent action are not possible.

JADE made it possible to avoid drawbacks found in MAS developed in MATLAB. The MAS developed in JADE can do peer-to-peer communication between agents in an efficient way. Chapter 4 presents the restoration of loads after isolation of faults using JADE and VTB, and the next section presents detection and isolation of the over-current fault in a radial distribution system, using MAS-developed JADE.

3.6 Agent Development in JADE for Fault Isolation

In this section the MAS is developed in JADE for fault detection and isolation and it is implemented on a test case.

3.6.1 Agent Development

MAS for fault isolation and detection consists of Switch Agent (SA) with several behaviors. Table 3.6 shows a short description of behaviors associated with an SA. A detailed description of behaviors, responsible for fault detection and isolation, associated with an SA is shown:

Table 3.6. Description of Behaviors associated with Switch Agent for Fault Isolation

Agents	Behaviors	Receive Message	Send Message/Signal	Description
Switch Agent	SWFaultQuery	-	- performative "QUERY_REF" and conversation ID "FAULT"	Sends message to all neighboring agents when the agent detects over-current.
	SWFaultReceive Reply	- performative "QUERY_REF" and conversation ID "FAULT"	- performative "INFORM" and conversation ID "FAULT"	Receives query from neighboring agent and replies with YES/NO if it has detected /not detected over-current.
	SWFaultAction	- performative "INFORM" and conversation ID "FAULT"	- performative "PROPOSE" and conversation ID "FAULT" - Sends OPEN signal to itself	Receives reply from neighboring agent and replies with OPEN if the neighboring agent has sent NO
	SWFaultReceive Action	- performative "PROPOSE" and conversation ID "FAULT"	Sends OPEN signal to itself	Receives reply from neighboring agent and sends OPEN to itself if neighboring agent has sent OPEN.

1. SWFaultQueryBehavior: In this behavior the agent sends message with performative "QUERY_REF" and Conversation ID "FAULT" to its downstream neighbors when it detects an over-current.
2. SWFaultReceiveReplyBehavior: In this behavior the agent receives message with performative "QUERY_REF" and Conversation ID "FAULT" and replies with performative "INFORM" and Conversation ID "FAULT". The content of the message sent is "yes"/ "no" if the agent detected/not-detected over-current.
3. SWFaultActionBehavior: In this behavior the agent receives message with performative "INFORM" and Conversation ID "FAULT". It sends message to its downstream neighbor with content OPEN if the downstream neighbor has replied "no" in (2) above. The performative of the message sent is "PROPOSE" and

Conversation ID “FAULT”. It also sends “open” to itself if the downstream neighbor has replied “no” in (2) above.

4. SWFaultReceiveActionBehavior: In this behavior the agent receives message with performative “PROPOSE” and Conversation ID “FAULT”. If the content of the received message is “open” then it sends “open” signal to itself.

3.6.2 Results

Figure 3.17 shows a VTB schematic of the test system, and Figure 3.18 shows a one-line representation of the system. Table 3.7 gives the data for the system. For the fault at F2, Switch Agents 2 and 3 detect the fault based on a directional over-current fault detection criterion. Figure 3.19 (a) shows the fault current and source current of one of the phases along with control signals from the agents. A delay has been incorporated into the algorithm to represent communication delay time and breaker opening time. Figure 3.19 (b) shows the complete set of messages between the agents for the process of detection and isolation of fault. Figure 3.20 is a snapshot of the command window showing the initialization and messages between agents during the fault isolation process.

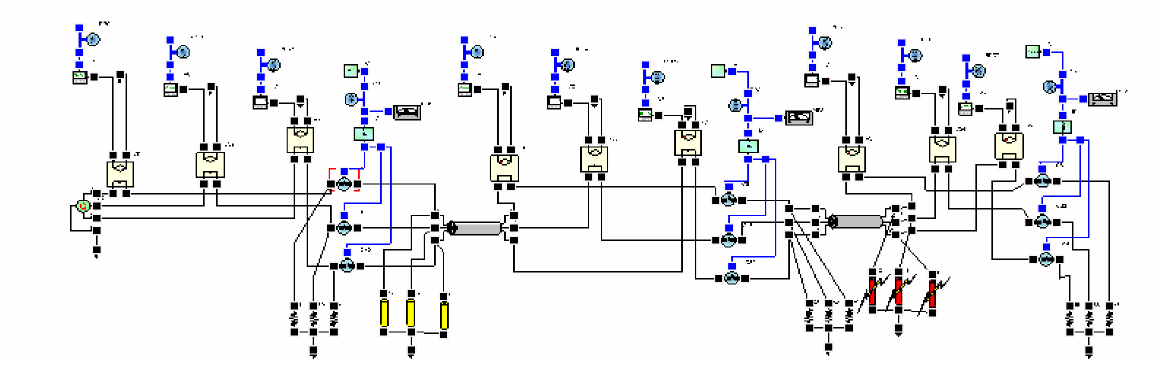


Figure 3.17 Test System II

Table 3.7. Test System II Parameters

Elements	Sub-Station A
3-Phase AC Voltage Source	169.7 Volt Peak
Load 1,2,3	10 Ohm/Phase
Cables	100 meter with Default Parameters in VTB

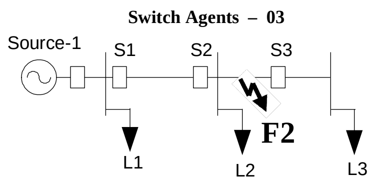
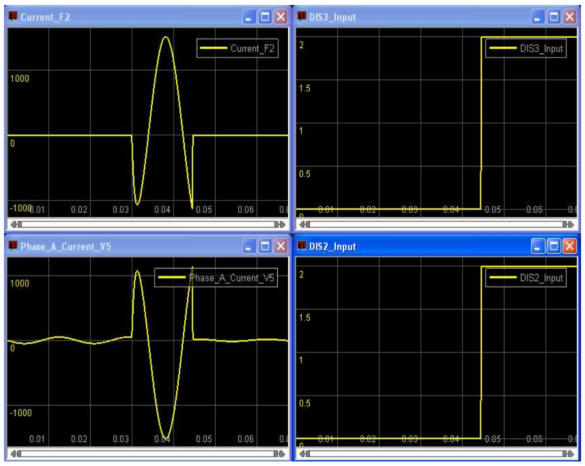
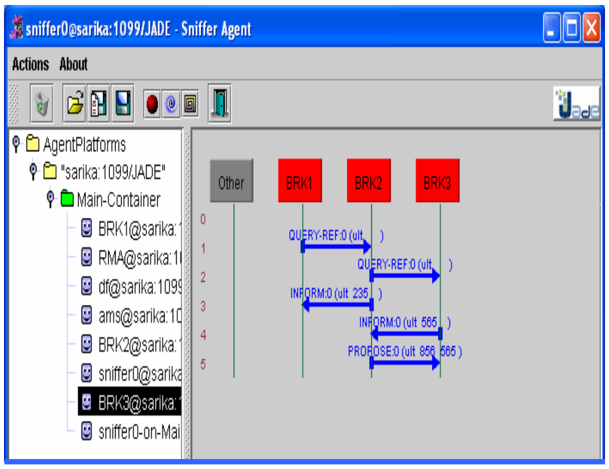


Figure 3.18 One line of Test System



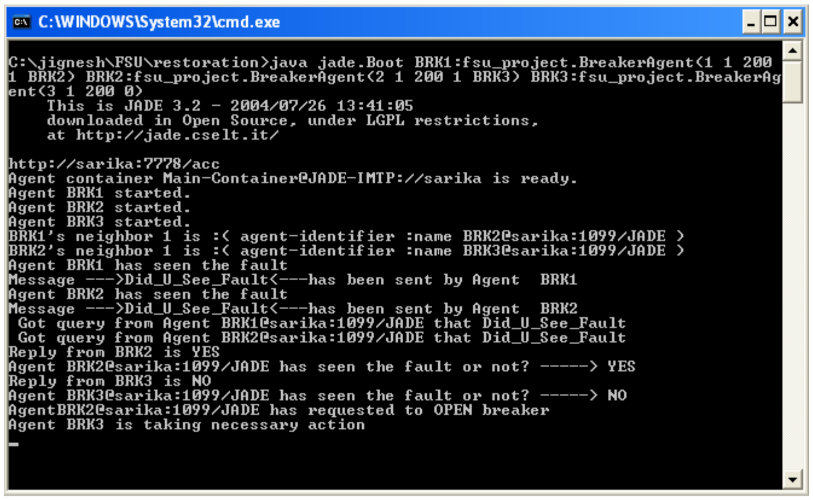
(a)



0
1 QUERY-REF:0 (ult)
2 QUERY-REF:0 (ult)
3 INFORM:0 (ult 235)
4 INFORM:0 (ult 565)
5 PROPOSE:0 (ult 856 565)

(b)

Figure 3.19 (a) Currents and Control Signals (b) Messages between Agents



```
C:\jignesh\FSU\restoration>java jade.Boot BRK1:fsu_project.BreakerAgent(1 1 200
1 BRK2) BRK2:fsu_project.BreakerAgent(2 1 200 1 BRK3) BRK3:fsu_project.BreakerAg
ent(3 1 200 0)
    This is JADE 3.2 - 2004/07/26 13:41:05
    downloaded in Open Source, under LGPL restrictions,
    at http://jade.cselt.it/

http://sarika:7778/acc
Agent container Main-Container@JADE-IMTP://sarika is ready.
Agent BRK1 started.
Agent BRK2 started.
Agent BRK3 started.
BRK1's neighbor 1 is :( agent-identifier :name BRK2@sarika:1099/JADE )
BRK2's neighbor 1 is :( agent-identifier :name BRK3@sarika:1099/JADE )
Agent BRK1 has seen the fault
Message --->Did_U_See_Fault<---has been sent by Agent  BRK1
Agent BRK2 has seen the fault
Message --->Did_U_See_Fault<---has been sent by Agent  BRK2
 Got query from Agent BRK1@sarika:1099/JADE that Did_U_See_Fault
 Got query from Agent BRK2@sarika:1099/JADE that Did_U_See_Fault
Reply from BRK2 is YES
Agent BRK2@sarika:1099/JADE has seen the fault or not? ------> YES
Reply from BRK3 is NO
Agent BRK3@sarika:1099/JADE has seen the fault or not? ------> NO
AgentBRK2@sarika:1099/JADE has requested to OPEN breaker
Agent BRK3 is taking necessary action
_
```

Figure 3.20 Snapshot of Command Window

3.7 Conclusions

This chapter formulates the restoration problem mathematically and proposes and implements a MAS solution. The MAS was developed in MATLAB, and the power system was simulated in VTB. A MATLAB-VTB co-simulation was achieved to implement the actions of agents in the power system. The development of MAS in MATLAB had some drawbacks, and hence JADE was investigated as a probable solution. Fault isolation was achieved using agents developed in JADE in this chapter. Chapter 4 shows the restoration using the MAS developed in JADE. Chapter 5 includes DG in the system and achieves load shedding to maintain the island formed as a result of fault isolation.

CHAPTER IV

RESTORATION USING MULTI-AGENT SYSTEM

4.1 Introduction

In this chapter the VTB simulation schematic is built for the test cases. The schematic displays the loads, generations, impedances, measuring devices, RPCs and controls. To reduce the number of blocks, hierarchical entities were built. The hierarchical entity constitutes the blocks forming a subsystem. This hierarchical representation also aids the visualization of related elements as one entity. Apart from simulation entities, software entities called agents also need to be developed. Three types of agents—Switch Agent (SA), Generator Agent (GA) and Load Agent (LA)—are developed for the restoration. An agent can be added or deleted in runtime without affecting the entire architecture. Thus, agents are very flexible; this flexibility facilitates the desired performance. Multiple SAs, LAs and GAs occur in the restoration package. The majority of the work each agent does is embedded in behaviors, which are methods that tell the agent how to react in a particular situation. Several behaviors have been developed for each agent. The implementation of the behavior is not the same for every agent of that type. The behaviors are extensions of the standard behavior classes of JADE. The test systems demonstrate that each agent can respond to the specific system conditions prevailing at that instant to achieve the desired goal.

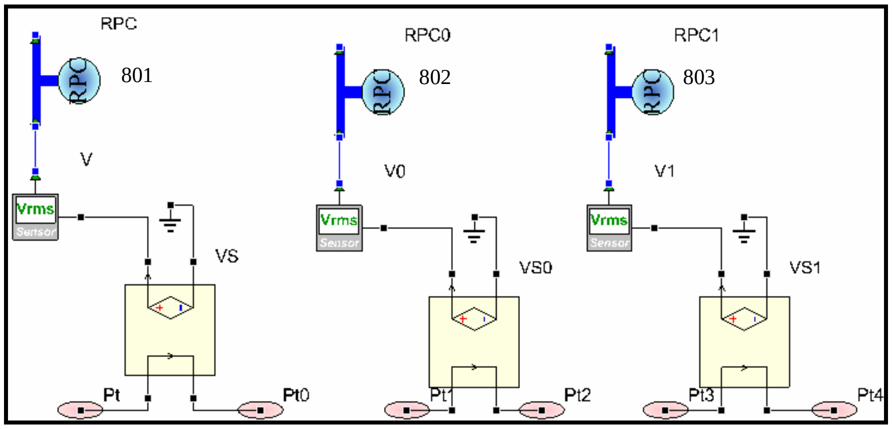
The knowledge is not concentrated on one agent, and one agent cannot make the decision in its entirety. The decision is collectively taken by a group of agents forming Multi-Agent System (MAS). Restoration strategies must change as the loading changes. This change of strategies is not possible in a centralized solution. For making real-time decisions, the intelligence needs to be embedded at the component level. Agents can adapt to the changing environment. The development of agents, as described in this chapter, first identifies the type of agents, the attribute initializations, then the conversations, the conversation language and finally the implementation. The results of a restoration with developed MAS show the information exchange between agents is restricted essential data resulting in a quick and efficient solution.

4.2 Hierarchical Models

Several hierarchical models were built in VTB in order to simplify the schematic because of its complexity due to larger system size. A hierarchical model helps to keep related elements together and divides the system into layers. Several hierarchical entities were built, namely:

4.2.1 Three-Phase Current Measurement

This hierarchy was built using a Current Controlled Voltage Source (CCVS) block in VTB, as VTB has no rms current measurement. CCVS converts the currents into rms voltage (representative of current as the gain of the block is set to unity). Agents access these currents through the RPC. In Figure 4.1, the RPC identity is set as 801-803 for independent current measurement of three phases of GAs to calculate the remaining capacity of the generator from the given maximum capacity. If, more than one GA is present in the system, then RPC ID can be increased to 901-903, 1001-1003 and so forth.



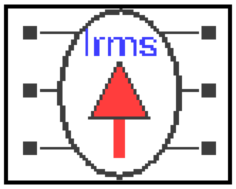
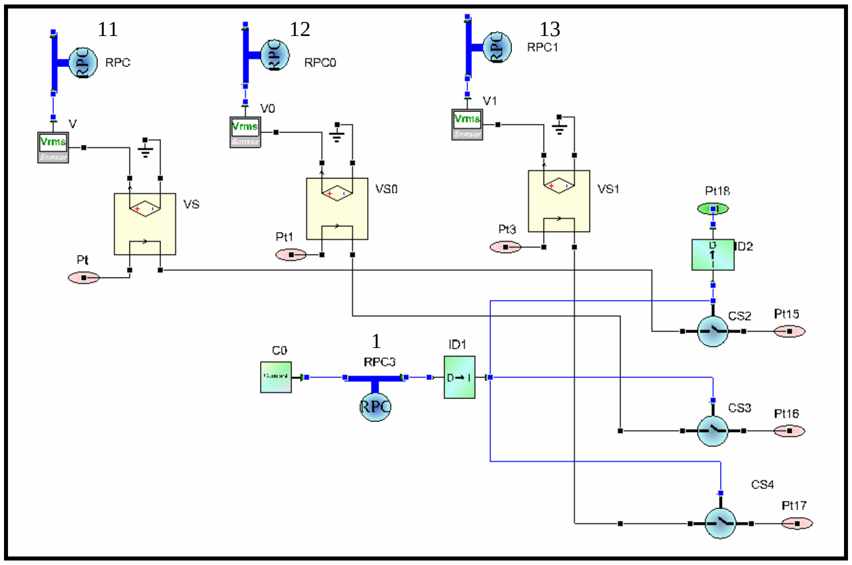


Figure 4.1 Hierarchical Model of Three-Phase Current Measurement with Icon

4.2.2 Three-Phase Current Measurement with Controllable Switch

Apart from the three-phase current measurement, a switch controlled by signals from the RPC is used. A graphical display of an RPC signal is set as viewable through the signal port outside of the hierarchical entity. In Figure 4.2, the RPC identity is set as 1 for the controllable switch of SAs to control the status of the switch. RPC identities of 11-13 are used by the SA to detect the current under abnormal conditions in order to start communications. If, more than one SA is present in the system, the RPC ID can be increased to 2 and 21-23, 3 and 31-33, and so forth.



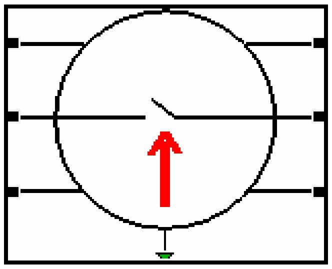
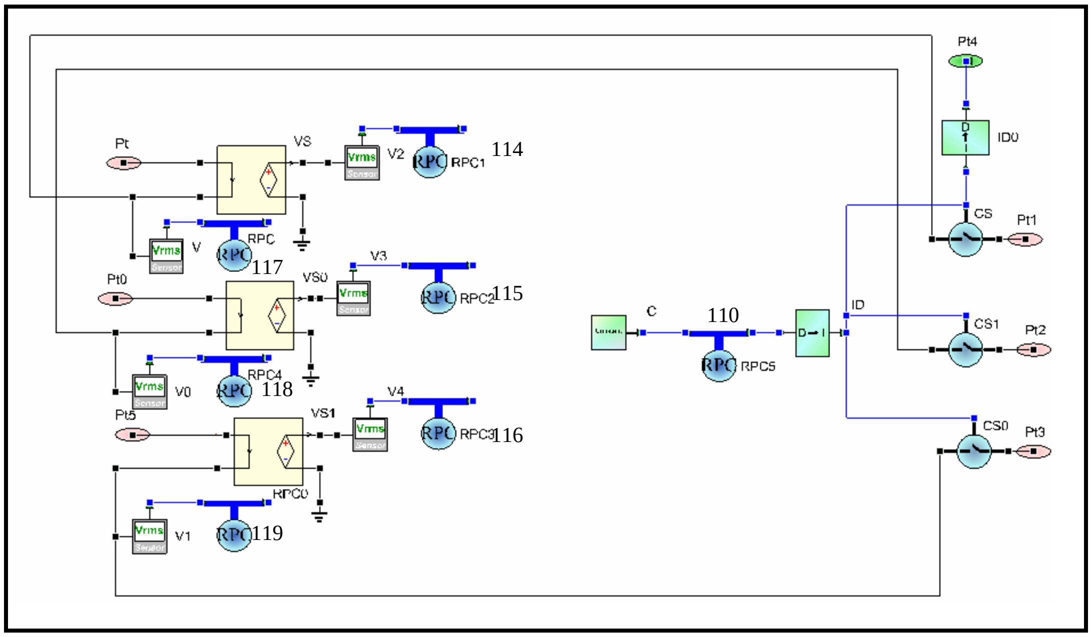


Figure 4.2 Hierarchical Model of Three-Phase Current Measurement with Controllable Switch with Icon

4.2.3 Three-Phase Current and Voltage Measurement with Controllable Switch

Apart from the three-phase current measurements and controllable switch, voltages are obtained by the LAs using the rms voltage measurement through the RPC. In Figure 4.3, the RPC identity is set as 110 for the controllable switch of LAs to supply or shed the load. RPC identities of 114-116 correspond to measurements of currents flowing into the load; identities of 117-119 correspond to measurements of voltages at the terminals of the load. If, more than one load agent is present in the system, then RPC ID can be increased to 120, 124-126 and 127-129, 130, 134-136 and 137-139 and so forth.



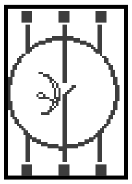


Figure 4.3 Hierarchical Model of Three-Phase Current and Voltage Measurement with Controllable Switch with Icon

4.3 Agent Algorithms

Three types of agents are in the proposed multi-agent system: SAs, LAs and GAs. The agents only communicate with neighboring agents. The actual work that an agent does is carried out in behaviors. This section describes the algorithmic flow of the different agents developed.

4.3.1 Switch Agent

Switch Agent is first initialized as:

SA (ID, Status, Rated Current, NumDnSWNeighbors, ID_DnSWNeighbors, NumUpSWNeighbors/NumUpGenNeighbors, ID_UpSWNeighbors/ID_UpGenNeighbors, NumDnLoadNeighbors, ID_DnLoadNeighbors, NumUpLoadNeighbors, ID_UpLoadNeighbors, Priority)

where,

ID: Agent Identification Number

Status: Open/Close

Rated Current: Maximum current

NumDnSWNeighbors: Number of downstream switch neighbors

ID_DnSWNeighbors: Agent Identification Number of downstream switch neighbors

NumUpSWNeighbors/NumUpGenNeighbors: Number of upstream switch/generator neighbors

ID_UpSWNeighbors/ID_UpGenNeighbors: Agent Identification Number of upstream switch/generator neighbors

NumDnLoadNeighbors: Number of downstream load neighbors

ID_DnLoadNeighbors: Agent Identification Number of downstream load neighbors

NumUpLoadNeighbors: Number of upstream load neighbors

ID_UpLoadNeighbors: Agent Identification Number of upstream load neighbors

The SA receives the measured voltages and currents and checks for over-current. If, it detects over current, it sends a message to neighboring agents for fault isolation.

After the fault is isolated, if there are loads that are left unsupplied then LAs start the restoration process. The SAs receive this restoration query, and they communicate with other agents to find a restoration path. If the number of available restoration paths is more than one, then the best path is chosen based upon the best available capacity value. If the available capacity is greater than the de-energized loads, then the SA sends a “close” message to the LA. If the available capacity is less than the de-energized loads, then the SA sends an “open” message to the LA. Figures 4.4 and 4.5 show the flowchart and algorithmic flow for a single SA.

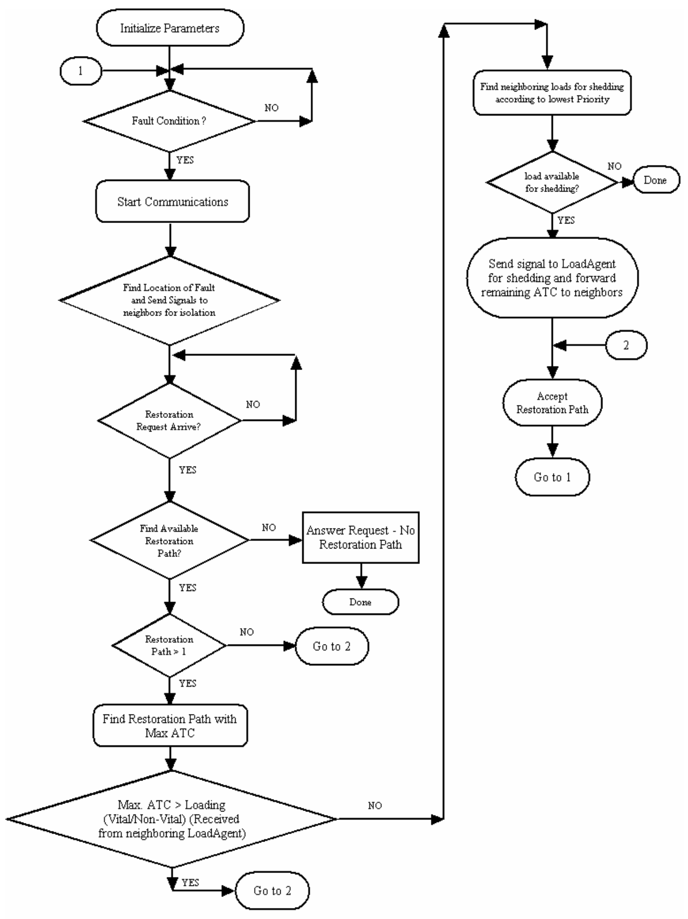

Initialize Parameters
1
Fault Condition ?
NO
YES
Start Communications
Find Location of Fault and Send Signals to neighbors for isolation
Restoration Request Arrive?
NO
YES
Find Available Restoration Path?
NO
Answer Request - No Restoration Path
Done
YES
Restoration Path > 1
NO
Go to 2
YES
Find Restoration Path with Max ATC
Max. ATC > Loading (Vital/Non-Vital) (Received from neighboring LoadAgent)
NO
YES
Go to 2
Find neighboring loads for shedding according to lowest Priority
load available for shedding?
NO
Done
YES
Send signal to LoadAgent for shedding and forward remaining ATC to neighbors
2
Accept Restoration Path
Go to 1

Figure 4.4 Flow Chart - Single Switch Agent

```
ALGORITHM for Switch Agent (SWAgent)

Setup()

Initialize all parameters

Set Agents $N_{DS}$, $N_{US}$ and $N_{LA}$
$N_{DS}$: Number of Downstream Agents
$N_{US}$: Number of Upstream Agents
$N_{LA}$: Number of Load Agents
FaultCount(rpcIDA, rpcIDB, rpcIDC)
{
  Read Data
  CurrentPhaseA ← getAPhaseI (ID)
  CurrentPhaseB ← getBPhaseI (ID)
  CurrentPhaseC ← getCPhaseI (ID)

  VoltagetPhaseA ← getAPhaseV (ID)
  VoltagePhaseB  ← getBPhaseV (ID)
  VoltagePhaseC  ← getCPhaseV (ID)

  Data Analyze;
  Data Storing;
}
FaultQueryBehavior

  $CurrentPhase_i$ = faultcount(recIDA, rpcIDB, rpcIDC)

  if ($CurrentPhase_i$ >5*$RatedCurrent_i$ ) //
    {
    SendMsg ($N_{Us}$+$N_{Ds}$, QUERY_REF, FAULT)
    }
FaultReceiveReplyBehavior

getMsg(QUERY_REF, FAULT)
replyMsg(content)//content YES or NO

FaultActionBehavior

if (content==”NO”)
{
    SendMsg (Sender, PROPOSE, FAULT, “OPEN”) //
    Soapset (rpcID,VALUE) //Fault Isolation
  }

FaultReceiveActionBehavior

getMsg(PROPOSE, FAULT)
if (content==”OPEN”) {
Soapset (rpcID,VALUE) //Fault Isolation }
Agent.addBehaviour(PreRestoreBehavior)

PreRestoreBehavior

   SendMsg ($N_{DS}$+ $N_{LA}$, REQUEST, PRERESTORE)
PreRestoreForwardBehavior

    getMsg (REQUEST, PRERESTORE)
   if (SWPriority == 0)
```

```
ContinueReceiveRestoreBehavior

  getMsg (REQUEST, RESTORE) PreFaultLoading, Status)

  if (SWPriority == 1 && NC){
    wait and receive all Msgs
    Request_Queue ← (AID, PreFaultLoading, Status )
    SendMsg($N_{LA}$+$N_{Ds}$, REQUEST, RESTORE, Request_Queue)}
  elseif (SWPriority == 0 && NC)
  SendMsg($N_{Ds}$, QUERY, LOADING)
  elseif (SWPriority ==2 && NC)
    Wait to Receive Msgs from Load if Present
    Request_Queue Queue ← (AID, PreFaultLoading, Status )
    SendMsg($N_{LA}$+$N_{Ds}$, REQUEST, RESTORE, Request_Queue)
  else //NO
    getMsg(REQUEST, RESTORE, Request_Queue)
    SendMsg($N_{US}$+$N_{Ds}$+ AIDMsg , QUERY, ATC)
ATCQueryBehavior

getMsg(QUERY, ATC)
SendMsg($N_{US}$, QUERY, ATC)

ATCInformBehavior

getMsg(INFORM, ATC, ATCValue)
if (SW NO){
  SendMsg($N_{US}$+$N_{DS}$-AIDMsg, INFORM, Aval_ATC, ATCValue)}
else{
  SendMsg($N_{DS}$, INFORM, ATC, ATCValue)}

ATCReceiveBehavior

if (SWPriority == 1){
  getMsg(INFORM, Aval_ATC, ATCValue)
  if (PreFaultViltalLoad Exist){
    Supply = Aval_ATC – PreFaultLoading(self_Viltal;
      if(Supply > 0)
      SendMsg((AIDPreFaultLoad (Vital, INFORM, Aval_ATC, ATCValue, Supply)}
  Supply_1 = Aval_ATC – PreFaultLoading(Viltal) – PreFaultLoading(Self);
  if(Supply_1 > 0)
    SendMsg((AIDPreFaultLoad(self), INFORM, Aval_ATC, ATCValue, Supply_1)
  Suppy_2 = Aval_ATC – PreFaultLoading(all)
  if(Supply_2 > 0)
    SendMsg(($N_{US}$, INFORM, Aval_ATC, ATCValue)}

if (SWPriority == 0){
  getMsg(INFORM, Aval_ATC, ATCValue, Supply)
 if  (Multiple ATC Received)
  Choose Best ATC
   if (supply <= 0)
     Soapset (rpcID,VALUE)}

if (SWPriority == 2){
  getMsg(INFORM, Aval_ATC, ATCValue)
  if  (Multiple ATC Received)
    Choose Best ATC
  Supply = AvalATC – PreFaultLoadign(Vital)
   if(Supply > 0){
     if((Aval_ATC – PreFaultLoading(self)) > 0)
       SendMsg(INFORM, Aval_ATC, Supply)
     else
       Soapset (rpcID,VALUE)}}
```

Figure 4.5 Algorithm for Switch Agent

4.3.2 Load Agent

The load Agent is first initialized as:

LA (ID, Priority, NumDnSWNeighbors, ID_DnSWNeighbors, NumUpSWNeighbors/NumUpGenNeighbors, ID_UpSWNeighbors/ID_DnSWNeighbors)

where,

Priority: Vital/Non-Vital and other parameters are same as SA.

The LA receives measured voltages and currents, and it checks for loss of voltage. If it detects loss of voltage, it sends message to neighboring switch agents for restoration. If it receives several replies to the restoration query, then it selects the one with maximum available capacity. If it receives a load-shedding request, then it sends “open” to itself. Figures 4.6 and 4.7 show the flowchart and algorithmic flow of a single LA.

4.3.3 Generator Agent

Generator Agent is first initialized as:

GA (ID, Capacity, NumDnSWNeighbors /NumDnLoadNeighbors, ID_DnSWNeighbors/ID_DnLoadNeighbors)

where

Capacity: Maximum rated capacity of the generator and the other parameters are same as SA.

The GA gets measured voltages and currents and calculates the remaining capacity after receiving the restoration request.

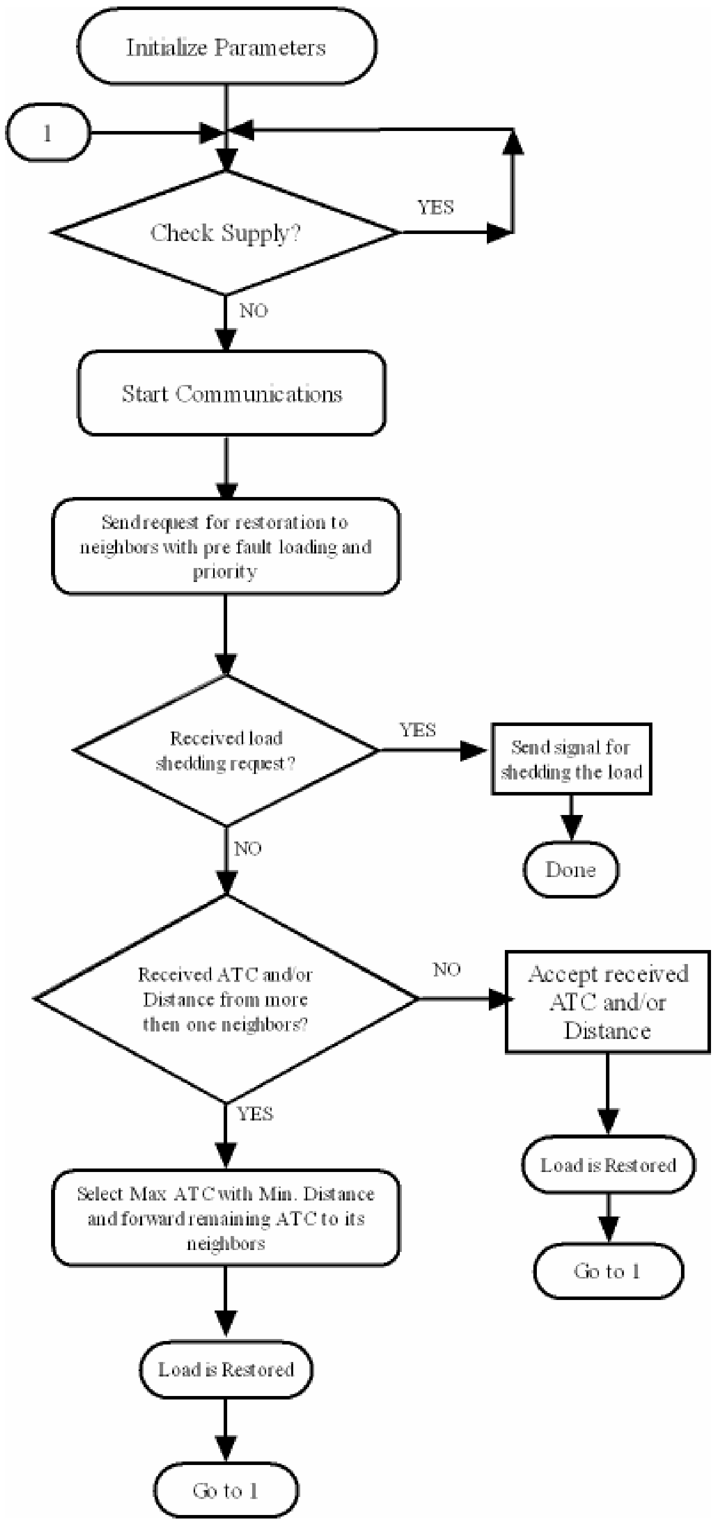

Initialize Parameters
1
Check Supply?
YES
NO
Start Communications
Send request for restoration to neighbors with pre fault loading and priority
Received load shedding request?
YES
Send signal for shedding the load
Done
NO
Received ATC and/or Distance from more then one neighbors?
NO
Accept received ATC and/or Distance
Load is Restored
Go to 1
YES
Select Max ATC with Min. Distance and forward remaining ATC to its neighbors
Load is Restored
Go to 1

Figure 4.6 Flow Chart - Single Load Agent

```
ALGORITHM for Load Agent (Load
Agent)

Setup()

Initialize all parameters
Set Agents $N_{DS}$ and $N_{US}$
$N_{DS}$: Number of Downstream Agents
$N_{US}$: Number of Upstream Agents
bestATC = max;
bestDist = max;
bestATCOffer = NullMsg;

PreFaultLoading()

  Read Data
  CurrentPhaseA ← getAPhaseI (ID)
  CurrentPhaseB ← getBPhaseI (ID)
  CurrentPhaseC ← getCPhaseI (ID)

  VoltagetPhaseA ← getAPhaseV (ID)
  VoltagePhaseB  ← getBPhaseV (ID)
  VoltagePhaseC  ← getCPhaseIV (ID)

  Data Storing;

*RestorationStartBehavior*

getMsg (REQUEST, PRERESTORE)
  SendMsg ($N_{DS}$, REQUEST, RESTORE,
PRE_FAULT  LOADING, Vital/Non-
Vital Status)


*ReceiveContinueReceiveRestoreStartBeh
avior*

getMsg (REQUEST, RESTORE)
SendMsg ($N_{DS}$+$N_{LA}$, REQUEST,
RESTORE, PreFaultLoading, Status)
```

```
*ATCInformBehavior*

getMsg(QUERY, ATC)
SendMsg ($N_{US}$, QUERY, ATC)

*ATCInformReceiveBehavior*

getMsg(INFORM, ATC)
  Receive Msg
  if (offer > bestATC)
    bestATC = offer;
    bestATCOffer = Msg;
    bestDistCost = offerDist;
  esleif (offer == bestATC)
    if (offerDist < bestDistCost)
      bestATC = offer;
      bestDistCost = offerDist;
     end
  end
  CheckList and Find Minimum Loading,
Vital/Non-Vital    Loading

  if ( bestATC – VitalLoading –
PreFaultLoading(self) > 1)
    Available to UpStream Unserved Load
bestATC =   bestATC – PreFaultLoading;
    SendMsg ($N_{US}$, INFORM, ATC,
bestATC)
  eslseif  (bestATC – VitalLoading > 1)
    SendMsg ( Self, LoadShedding)
    SendMsg ($N_{US}$, INFORM, ATC,
bestATC)
  end

  if (MinLoading < ATC <
PreFaultLoading(self) OR  MinLoading <
ATC < Vital/Non-VitalLoadingInList)
    SendMsg ($N_{US}$, INFORM, ATC,
bestATC)
    SendMsg ( Self, LoadShedding)
  end
```

Figure 4.7   Algorithm for Load Agent

4.4 Agent Development

An agent has several behaviors in which the tasks it must perform are embedded. Analyzing the concepts involved in restoration leads to the behavior architecture. Table 4.1 shows the short description of behaviors associated with different agents. A detailed description of behaviors, responsible for restoration and load shedding if necessary, associated with each agent is as below.

4.4.1 Switch Agent

1. SWRestoreBehavior: This behavior corresponds to step-I in Figure 4.8. The agent receives a message with performative “QUERY_REF” and Conversation ID “RESTORE.” This message is extracted and saved, categorizing the content as type, name, status and loading. Type denotes whether the message is from a load agent or a switch agent, name denotes the unique agent identification; status refers to critical/vital or non-critical/non-vital, and loading is the pre-fault loading. The status of the received -

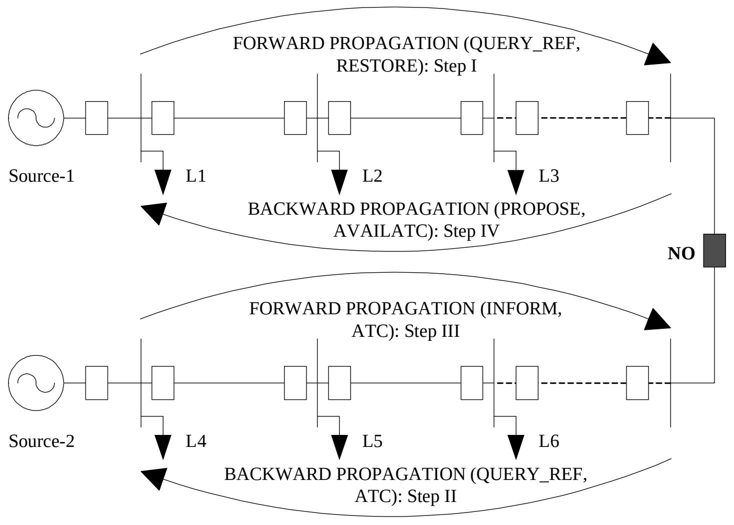

FORWARD PROPAGATION (QUERY_REF, RESTORE): Step I
Source-1
L1
L2
L3
BACKWARD PROPAGATION (PROPOSE, AVAILATC): Step IV
NO
FORWARD PROPAGATION (INFORM, ATC): Step III
Source-2
L4
L5
L6
BACKWARD PROPAGATION (QUERY_REF, ATC): Step II

Figure 4.8 Messages Flow for Restoration between Different Agents

Table 4.1. Description of Behaviors associated with Agents for Restoration without DG

Agents	Behaviors	Receive Message	Send Message/Signal	Description
Switch Agent	SWRestore	- performative “QUERY_REF” and conversation ID “RESTORE”	- performative “QUERY_REF” and conversation ID “RESTORE” - performative “QUERY_REF” and conversation ID “ATC”	Receives message from either load agent or switch agent and saves the message content in predefined format
	SWATC Forward	- performative “QUERY_REF” and conversation ID “ATC”	- performative “QUERY_REF” and conversation ID “ATC”	Saves sender’s unique agent ID and forwards the message
	SWATC Inform	- performative “INFORM” and conversation ID “ATC”	- performative “INFORM” and conversation ID “ATC” - performative “PROPOSE” and conversation ID “AVAILATC”	Receives the message from either generator agent or switch agent and forwards to the agent who asked for ATC
	SWAvail ATC	- performative “PROPOSE” and conversation ID “AVAILATC”	- performative “PROPOSE” and conversation ID “ATCREPLY” - performative “PROPOSE” and conversation ID “AVAILATC”	Does load shedding, load allocation, best ATC selection
	SWAvail ATCAction	- performative “PROPOSE” and conversation ID “ATCREPLY”	- Send OPEN/CLOSE signals to itself	Sends CLOSE/OPEN signal when content of the received message is ACCEPT/DECLINE
Load Agent	Load Restoration Start	-	- performative “QUERY_REF” and conversation ID “RESTORE”	Detects the loss of voltage and send message with pre-fault loading and critical/non-critical status of load
	LoadAvail ATCAction	- performative “PROPOSE” and conversation ID “AVAILATC”	- performative “PROPOSE” and conversation ID “ATCREPLY” - Send OPEN/CLOSE signals to itself	Sends CLOSE/OPEN signal when content of the received message is ACCEPT/DECLINE. Does best ATC selection when needed
Generator Agent	GenATCInform	- performative “QUERY_REF” and conversation ID “ATC”	- performative “INFORM” and conversation ID “ATC”	Calculates its remaining ATC from the current loading given maximum capacity

message can be “T”, which denotes the total loading of the switch that includes all its downstream loading including critical/vital and non-critical/non-vital. If there is already a saved message with status critical/vital of the same agent (that sent the message with status “T”), then this total loading is subtracted from the critical/vital loading and the remaining loading is saved with status non-critical/non-vital. Critical/Vital loading and non-critical/non-vital loading are separately saved and sorted in ascending order of their loading. A flag is assigned in case multiple switches occur at a node. The flag is set to one if no generator is present at the far end of the feeder containing the switch. More specifically, the switch with flag one sends the message with status “T”. An action is taken according to:

a. An agent with a normally closed switch, with flag zero, which has not sent the query even once, changes the type and name of the message content and forwards this message regardless of whether the status of the message received is critical/vital or non-critical/non-vital. This message is forwarded to downstream switch neighbors if it received the message from upstream switch neighbors and vice versa.

b. An agent with a normally closed switch, with flag zero, and which has sent the query at least once changes the type and name of the message content and forwards this message in case the status of the message received is critical/vital. This message is forwarded to downstream switch neighbors if it has received the message from upstream switch neighbors and vice versa. In case of non-critical/ non-vital status, the message is only saved as described in (1) above.

c. An agent with a normally closed switch, with flag one, and which has not sent the query even once changes the type, name, status ("T") and loading of the message content if the message received has non-critical/non-vital status, before the message is sent. If, the message received has critical/vital status, then apart from the total loading message, it also sends a critical/vital loading message that it received after changing the type and name. In either case, the message is sent to downstream switch neighbors if it has received message from upstream switch neighbors and vice versa.

d. An agent with a normally closed switch, with flag one, and which has sent the query at least once changes the type and name of the message content before sending, if the received message has critical/vital status. The message is sent to downstream switch neighbors if it has received a message from upstream switch neighbors and vice versa. In case of non-critical/non-vital status, the message is only saved as described in (1) above.

e. An agent with a normally open switch which has not sent the query even once changes the message content to ATC (Available Transfer Capacity) and forwards this message changing the conversation ID to "ATC" regardless of whether the status of the message received is critical/vital or non-critical/non-vital. This message is forwarded after some delay to downstream switch neighbors if it has received message from upstream switch neighbors and vice versa. This delay ensures the receiving of other restore queries. The unique agent identifier for this agent is stored.

f. An agent with a normally open switch, which has sent the query at least once, saves the message as described in (1) above regardless of whether the status of the message received is critical/vital or non-critical/non-vital.

2. SWATCForwardBehavior: This behavior corresponds to Step-II in Figure 4.8. It receives messages with performative “QUERY_REF” and Conversation ID “ATC” and takes no action when the switch agent has flag set to one. In the case of a switch agent with flag set to zero, it stores the unique agent identifier of the sender of the message and forwards to downstream switch neighbors and generator neighbors if it has received message from upstream switch neighbors and vice versa.

3. SWATCInformBehavior: This behavior corresponds to Step-III in Figure 4.8. It receives messages with performative “INFORM” and Conversation ID “ATC.” This agent forwards the message to the agents stored in SWATCForwardBehavior only when it is not the agent stored in SWRestoreBehavior point (e); otherwise, the message is forwarded with conversation ID “AVAILATC”, performative “PROPOSE” and content as ATC and remainingATC to the agents stored in SWRestoreBehavior. For an agent with a normally open switch, remainingATC is same as ATC.

4. SWAvailATCBehavior: This behavior corresponds to Step-IV in Figure 4.8. It receives a message with performative “PROPOSE” and Conversation ID “AVAILATC.” This message is extracted and saved, categorizing the content as ATC and remainingATC. Whenever the switch agent receives an ATC better than the previous one, it sends a message with content decline and conversation ID “ATCREPLY” to the previous one and stores the message of the agent that sent the better ATC. The content of

this stored message is ATC and remaining ATC. The remaining ATC is also saved as supplied ATC. After receiving messages from all the downstream agents, the following steps are performed:

a. If the remaining ATC is zero then the switch agent sends an open signal to itself; otherwise, it performs steps (b)-(e).

b. Starting with the first entry in the critical/vital loading saved in (1) above, it is checked if the remaining ATC is greater than any entry. If it is/is not and the load is the switch's own load, then it sends a message with content "close"/"open" and supplied ATC. The remaining ATC is now updated. If, the remaining ATC is/is not greater than any entry and the load is not the switch's own load, then the loading and name is saved/not saved in ForwardATC or the loading is added to ForwardATC corresponding to the name of that agent if the name already exists. The remainingATC is now updated.

c. Starting with the first entry in the non-critical/non-vital loading saved in (1) above, it is checked if the remaining ATC is greater than any entry. If it is/is not and the load is switch's own load then it sends message with content "close"/"open" and supplied ATC. The remaining ATC is now updated. If remaining ATC is/is not greater than any entry and the load is not the switch's own load, then the loading and name is saved as ForwardATC/UnsuppliedLoad or the loading is added to ForwardATC/UnsuppliedLoad corresponding to the name of that agent if the name already exists. The remaining ATC is now updated.

d. If there is still some remainingATC, then the unsupplied load list is searched for a switch with its flag set to one, i.e., a switch that has sent total loading, and remainingATC is added to the ForwardATC corresponding to this agent name. The total loading is preferred because the other switches send only the loading received by their neighboring load agent. If no priority switch occurs in the unsupplied load list, then remainingATC is added to the ForwardATC, corresponding to the agent name of any other entry of the unsupplied load list.

e. After steps (a)-(d) are completed, a message is sent to each agent name saved in ForwardATC with content suppliedATC and the value saved in ForwardATC.

5. SWAvailATCActionBehavior: This behavior also corresponds to Step-IV in Figure 4.8. It receives message with performative “PROPOSE” and Conversation ID “ATCREPLY.” The agent sends a “close”/“open” signal if the content is accept/decline, but a normally open switch closes after some time delay to ensure load shedding to avoid overloading.

4.4.2 Load Agent

1. LoadRestorationStartBehavior: This behavior corresponds to Step-I in Figure 4.8. It sends a message with performative “QUERY_REF,” Conversation ID “RESTORE” and content as load status (critical/vital or non-critical/non-vital) and pre-fault loading, when the load agent detects loss of voltage.

2. LoadAvailATCActionBehavior: This behavior corresponds to Step-IV in Figure 4.8. It receives message with performative “PROPOSE” and Conversation ID “AVAILATC.” This message is extracted and saved, categorizing the content as action and

remainingATC. In case the load agent receives this message from multiple switch agents, and if the ATC is better than the previous one, it sends a message with content decline changing the conversation ID "ATCREPLY" to the previous one and stores the message of the agent that sent the better ATC. The content of this stored message is action and remainingATC. After receiving messages from all the downstream switch agents the load agent sends an "open"/"close" signal if the action is decline/accept.

4.4.3 Generator Agent

GenATCInformBehavior: This behavior receives a message with performative "QUERY_REF" and Conversation ID "ATC." It calculates its remaining capacity by subtracting the current loading from the maximum capacity. In reply to this query, it sends a message with performative "INFORM" and content remaining capacity.

4.5 Results

A power system test case was constructed in VTB. Chapter 3 described the detection of a fault and restoration using a MATLAB agent environment. This section focuses on restoration using JADE for developing the multi-agent framework. The agents only communicate with neighboring agents. The MAS is tested on a system without DG for restoration of loads after the fault isolation. The high priority of critical/vital loads must be considered during restoration. In the one line diagram of all the test cases load breakers are not shown for simplicity.

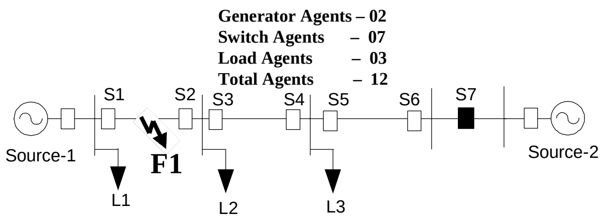


Figure 4.9 Distribution Network I

4.5.1 Test System I

The system in Figure 4.9 is a 5-node system with two generators and three loads; Load 2 is critical/vital, and Loads 1 and 3 are non-critical/non-vital. The rating of each load is 8.5 Amps/phase, and the generators in the model are ideal voltage sources with a positive sequence L-N rms as 120 Volts. The cables are 100 meters in length. In the simulation, fault 1 occurs at 0.1 seconds. Figure 4.10 shows the VTB schematic. The initialization of the MAS is shown in Appendix A.1.

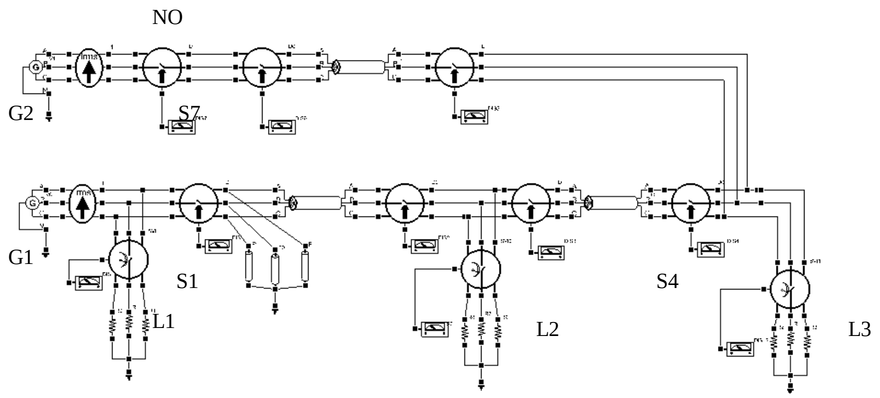


Figure 4.10 VTB Schematic of Distribution Network – I

This system is comprised of two generator agents, seven switch agents and three load agents. The initialization of one agent of each generator, load and switch is as:

- *GA (ID, Capacity, NumDnSWNeighbors /NumDnLoadNeighbors, ID_DnSWNeighbors/ID_DnLoadNeighbors)*

  G1: GA(8 100 2 S1 L1)

- *LA (ID, Priority (Vital/Non-Vital), NumDnSWNeighbors, ID_DnSWNeighbors, NumUpSWNeighbors/NumUpGenNeighbors, ID_UpSWNeighbors/ID_DnSWNeighbors)*

  L2: LA(110 V 1 S3 1 S2)

- *SA (ID, Status (Open (1)/Closed (0)), Rated Current, NumDnSWNeighbors, ID_DnSWNeighbors, NumUpSWNeighbors/NumUpGenNeighbors, ID_UpSWNeighbors/ID_UpGenNeighbors, NumDnLoadNeighbors, ID_DnLoadNeighbors, NumUpLoadNeighbors, ID_UpLoadNeighbors, Priority)*

  S2: SA(2 1 100 1 S3 1 S1 1 L2 0)

As seen, G1 has two neighbors, L2 has one downstream and one upstream neighbor with status vital, S2 has one downstream switch neighbor and one upstream switch neighbor along with one downstream load neighbor and no priority. After the isolation of the fault, Loads 2 and 3 are de-energized. This action leads to asynchronous messages with performative "QUERY_REF" and conversation ID "RESTORE" from L2 (Load Agent 2)-S3 (Switch Agent 3). The content of this message is "LLA2 V36.10" where "L" indicates message is sent by Load Agent, "LA2" is the name of this agent, "V" indicates the vital status and "36.10" is the pre-fault loading. Similar message is sent from L3-S5. The critical/vital loading information is passed from S3-S7 as shown in Messages 5-8 in Figure 4.11. The content of the message sent from S3-S4 is "SSW3

V36.10" where "S" indicates message is sent by Switch Agent, "SW3" is the name of this agent, "V" indicates the vital status of the loading received by this agent and "36.10" is the pre-fault loading received by this agent. Message 9 in Figure 4.11 is not forwarded to downstream switch agents, because S5 has already forwarded vital loading which saves communication. S7 changes the conversation ID to "ATC" and sends a message to G2 (Generator Agent 2) as shown in Message 10 in Figure 4.11. G2 changes the performative to "INFORM" and passes its remaining capacity "40.0" to S7 as shown in Message 11. S7, which is normally open, changes the performative to "PROPOSE" and conversation ID to "AVAILATC". The content of the message is "40.0 40.0" which indicates the available capacity received by the Switch Agent and the capacity it can supply to the respective Load Agent. S6 sends "ACCEPT" to S7 to indicate acceptance of the capacity. Similarly messages are passed from S6-S5 and S5-S4. S5 has the knowledge of vital Load 2 "36.10" due to Message 6. Since the capacity "40.0" is enough to supply only vital Load 2 and not Load 3 which is non-vital, S5 sends "OPEN" to L3. S3 sends "CLOSE" to L2 since the capacity received "40.0" is enough to supply L2. Figure 4.12 shows that Load 2 is restored in 0.3 seconds, incorporating a delay for physical constraints, and the current of Generator 2 is increased to 8.5 Amps. Figure 4.12 shows that the status of Switch 7 is set to "close" and Load 3 is shed.

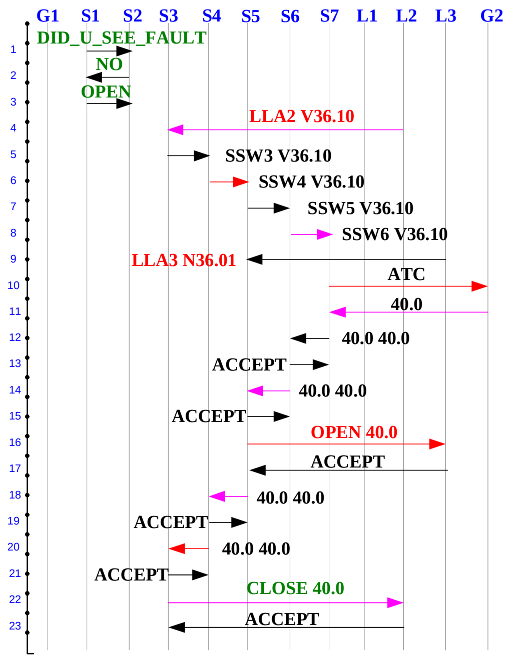


Figure 4.11 Trace of Messages between Agents for Partial Service Restoration

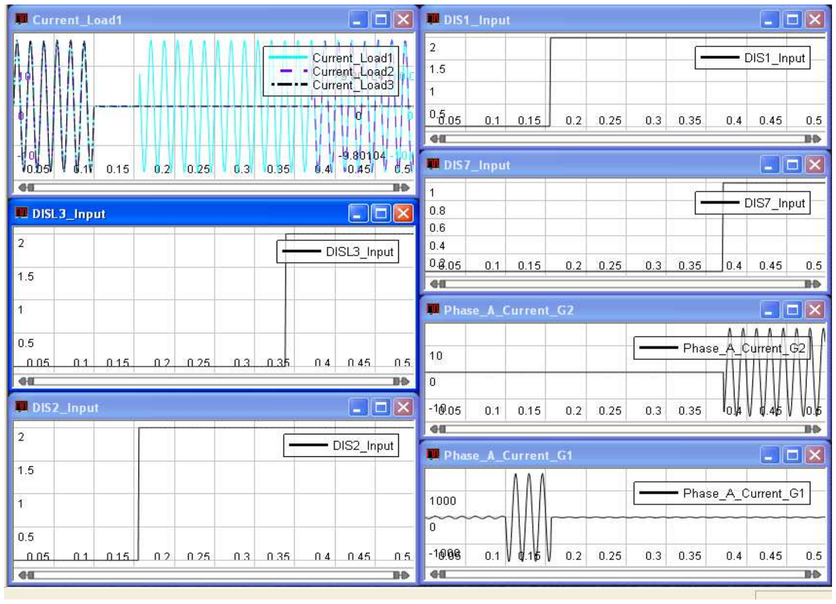


Figure 4.12 Currents: Load 1, Load 2 and Load 3, DISL3: Load 3 Control Signal, DIS2: Switch 2 control signal, DIS1: Switch 1 Control Signal, DIS7: Switch 7 Control Signal, Generator 2 Phase A Current, Generator 1 Phase A Current (starting with upper left corner and reading column-wise)

4.5.2 Test System II

Distribution networks as shown in Figure 4.13 were considered. The system is a 14-node system with two generators and ten loads with Loads 2, 10, 7 and 12 being critical/vital, and Loads 1, 3, 4, 5, 6, 8, 9 and 11 being non-critical/non-vital. The rating of each load is 12 Amps/phase with other parameters of the system the same as test system I. Figure 4.14 shows the VTB schematic of the system. Switch 34 is a normally open switch, and the rest are normally closed. The initialization of the MAS is shown in Appendix A.2.

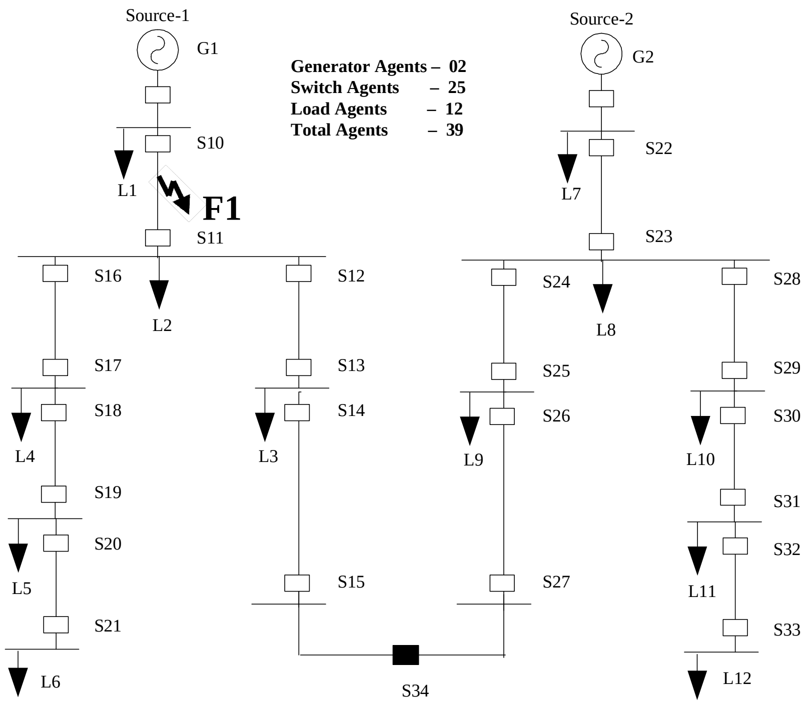


Figure 4.13 Distribution Network-II

A. Complete Service Restoration

To demonstrate complete service restoration, a fault F1 as shown in Figure 4.13, occurs in the simulation at 0.1 seconds. The capacity of each of the Sources 1 and 2 is 550 A. Chapter 3 gives the details of detection and isolation of faults. After isolation of the fault by opening Switches 10 and 11, Loads 2-6 are de-energized. This action leads to asynchronous messages with performative “QUERY_REF” and conversationID “RESTORE” from L2 (Load Agent 2)-S3 (Switch Agent 3), L3-S5, L4-S8, L5-S10 and L6-S12. The critical/vital loading information is passed from S21-S34 and from S12-S34, as shown in Messages 5-15 and Messages 20-23 in Figure 4.15.

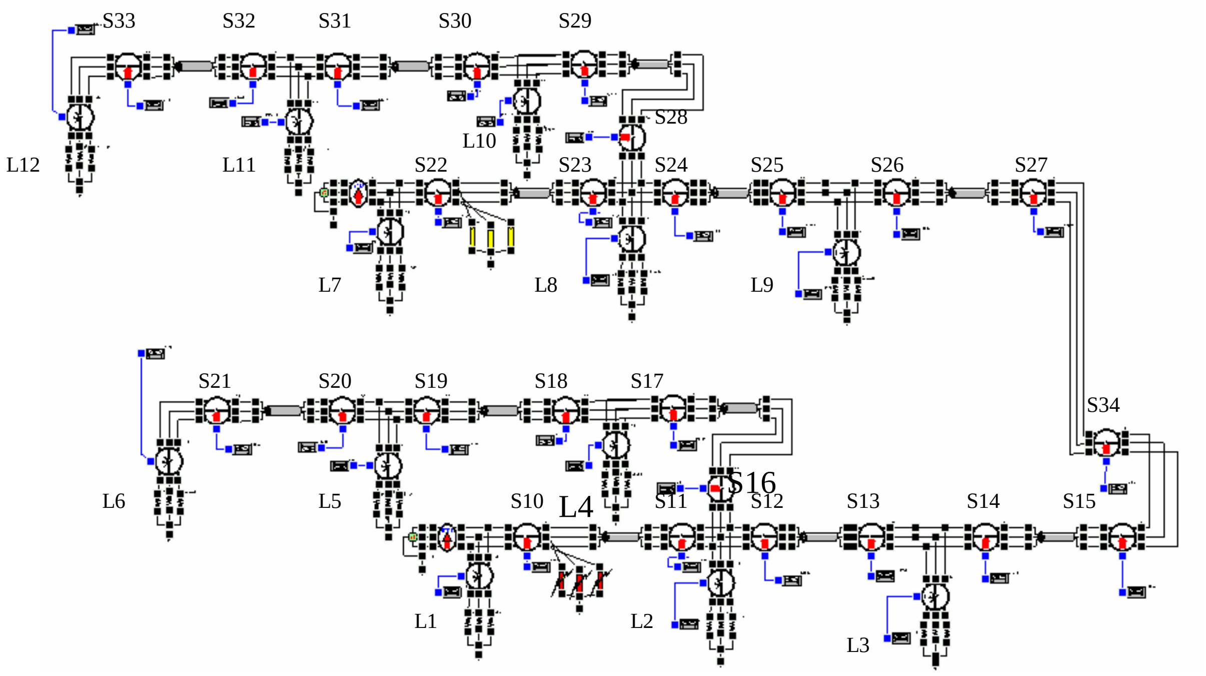


Figure 4.14 VTB Schematic of Distribution Network II

Message 19 in Figure 4.15 is not forwarded to downstream switch agents, because the S14 has already forwarded the vital loading, which saves communication. S34 changes the conversationID to “ATC” and sends a message to G2 (Generator Agent 2), as shown in Messages 24-30 in Figure 4.15. G2 changes the performative to “INFORM” and passes its remaining capacity to S34, as shown in Messages 31-37. S34 changes the performative to “PROPOSE” and conversationID to “AVAILATC,” and Loads 2-6 are restored, since the remaining capacity of G2 is enough to supply these loads, as shown in Messages 38-67 in Figure 4.15. Figure 4.16 shows that Loads 5 and 6 are restored in 0.037 seconds, not considering any physical time delay and the current of Source 2 is increased from 67A to 116A. Figure 4.17 shows that the status of Switch 34 is set to “close” (“1” indicates “close”, whereas “2” indicates “open”) and Loads 2, 3 and 4 are also restored.

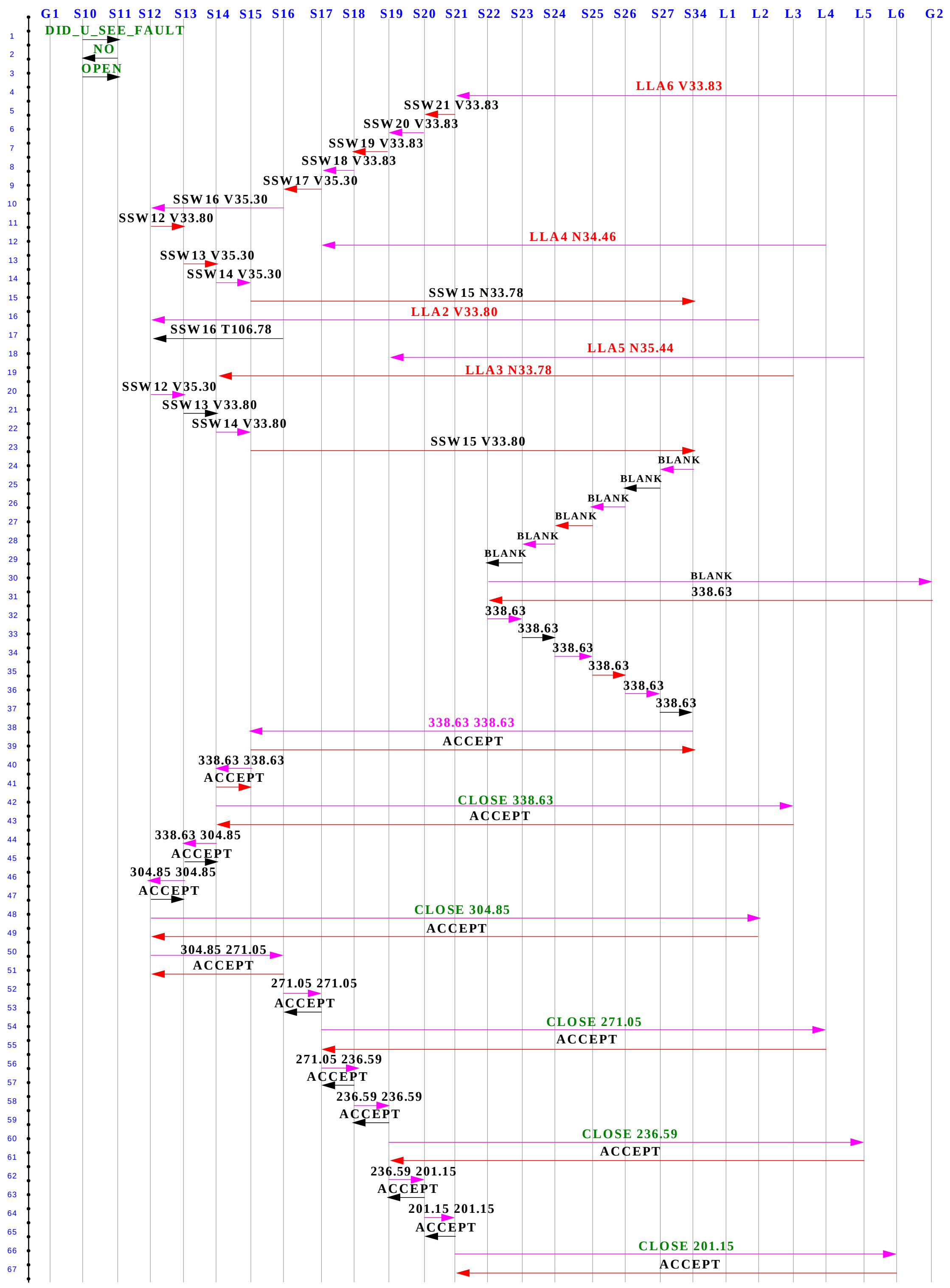


Figure 4.15 Trace of Messages between Agents for Full Service Restoration

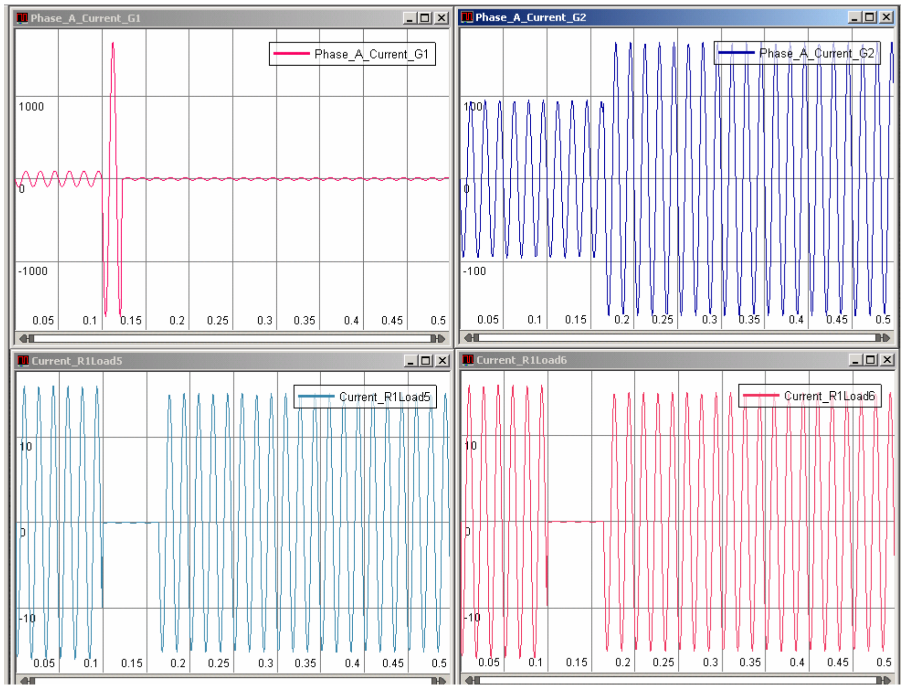


Figure 4.16 Source 1, Load 5, Source 2 and Load 6 Phase A Currents (starting with upper left corner and reading column-wise)

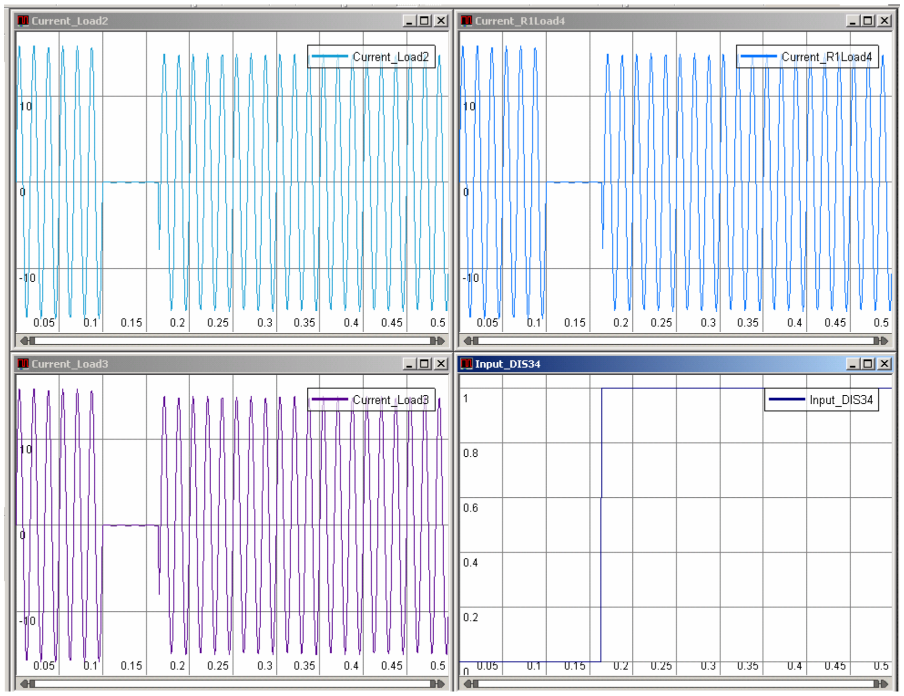


Figure 4.17 Load 2, Load 3, Load 4 Phase A Currents and Switch 34 Status (starting with upper left corner and reading column-wise)

B. Partial Service Restoration

The same fault, F1, as shown in Figure 4.13, was simulated at 0.1 seconds. The capacity of each of the Sources 1 and 2 is 350 A.

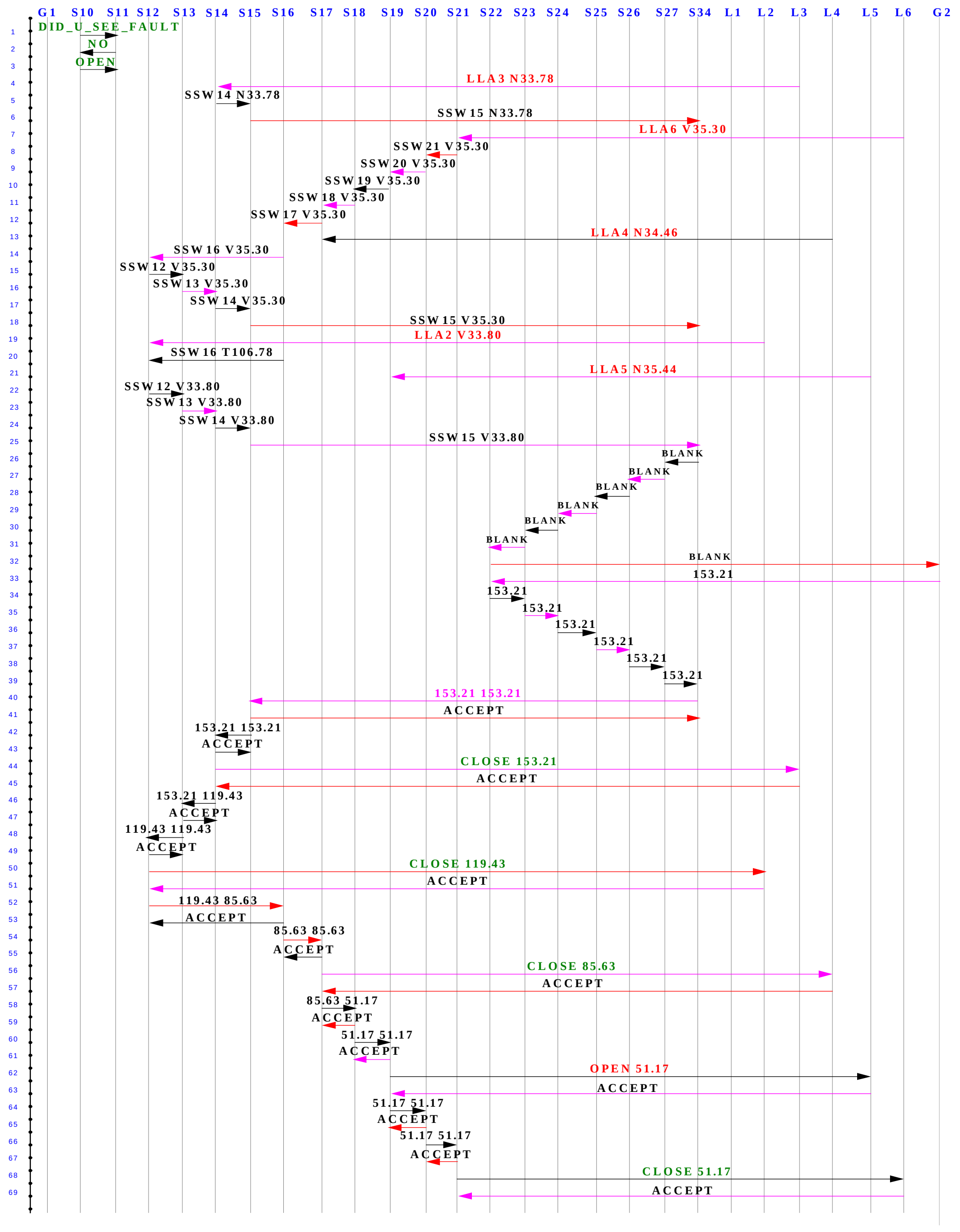


Figure 4.18 Trace of Messages between Agents for Partial Service Restoration

After the isolation of the fault by opening Switches 10 and 11, the Loads 2-6 are de-energized. Figure 4.18 shows a trace of the messages. Since the remaining capacity of Source 2 is not enough to supply all un-restored loads, Load 5 is shed because it is non-critical/non-vital and at the far end of restoration path. Load 6 is restored even though it is at the farthest end of the restoration path, since it is a critical/vital load.

Figure 4.19 shows that Load 6 is restored in 0.034 seconds and Load 5 is shed. Source 2's current is increased from 67A to 107A. Figure 4.20 shows that Loads 2, 3 and 4 are also restored.

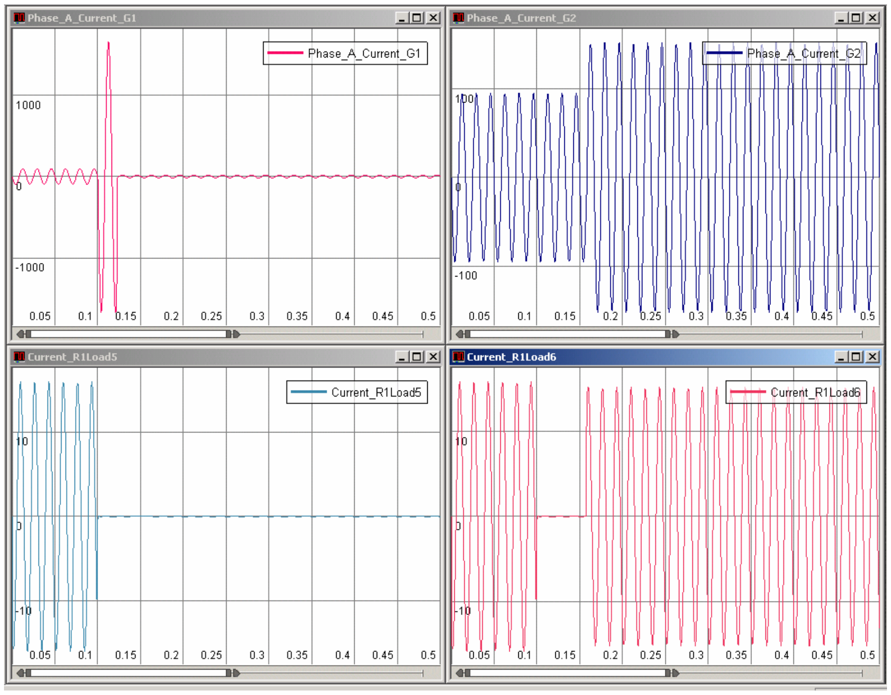


Figure 4.19 Source 1, Load 5, Source 2 and Load 6 Phase A Currents (starting with upper left corner and reading column-wise)

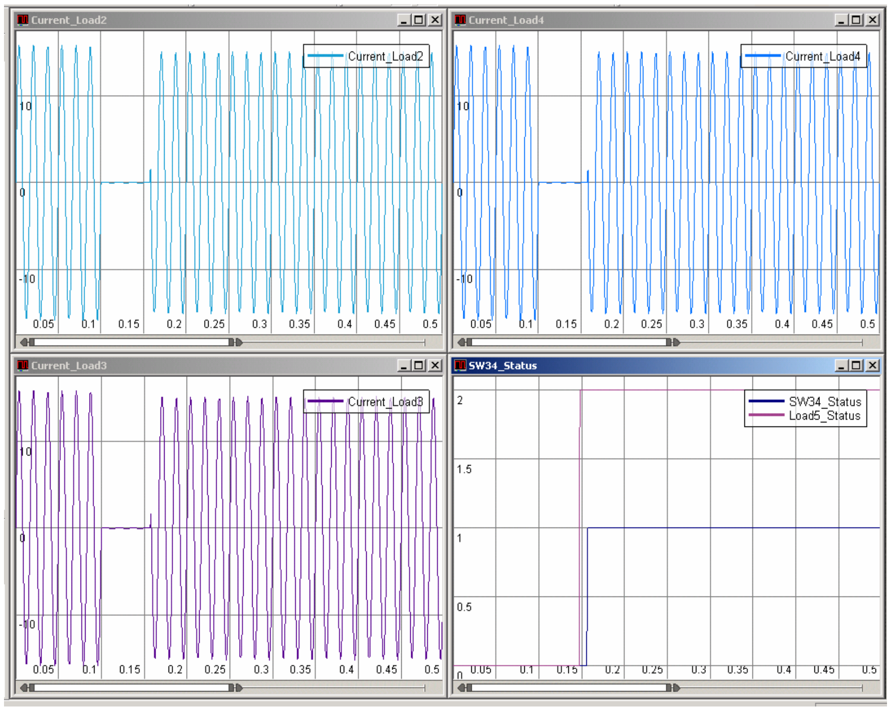


Figure 4.20 Load 2, Load 3, Load 4 Phase A Currents and Load 5, Switch 34 Status (starting with upper left corner and reading column-wise)

The status of the Switch 34 is set to close, and Load 5 status is set to open. Load 5 is shed before Switch 34 is closed, so that Source 2 is not overloaded.

4.5.3 Test System III

The system in Figure 4.21 is a 12-node system with four generators and eight loads; Loads 2, 4, 6 and 8 are critical/vital, and Loads 1, 3, 5, and 7 are non-critical/non-vital. The rating of Loads 1, 3, 5 and 7 is 12 Amps/phase, and the rating of Loads 2, 4, 6 and 8 is 8 Amps/phase. Switches 20, 24, 25, 26, 27 and 28 are normally open, and the rest are normally closed. Figure 4.22 shows the VTB schematic of the system. Switches 24-28 and Switch 20 are normally open switches; the rest are normally closed.

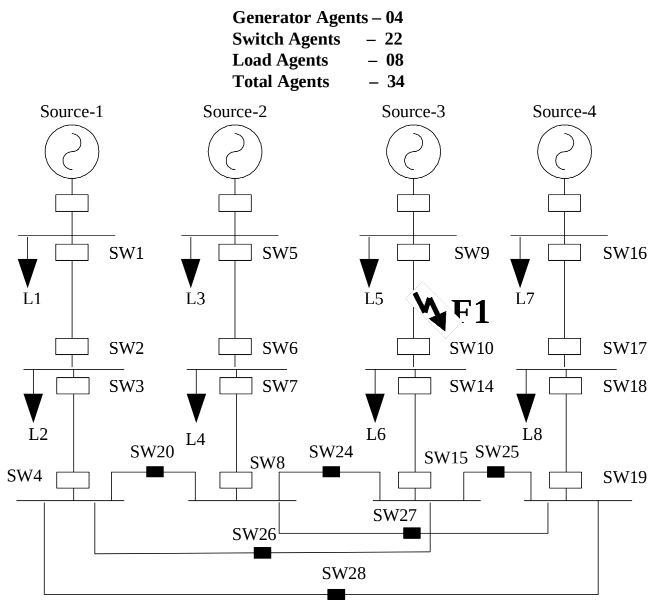


Figure 4.21 Distribution Network III

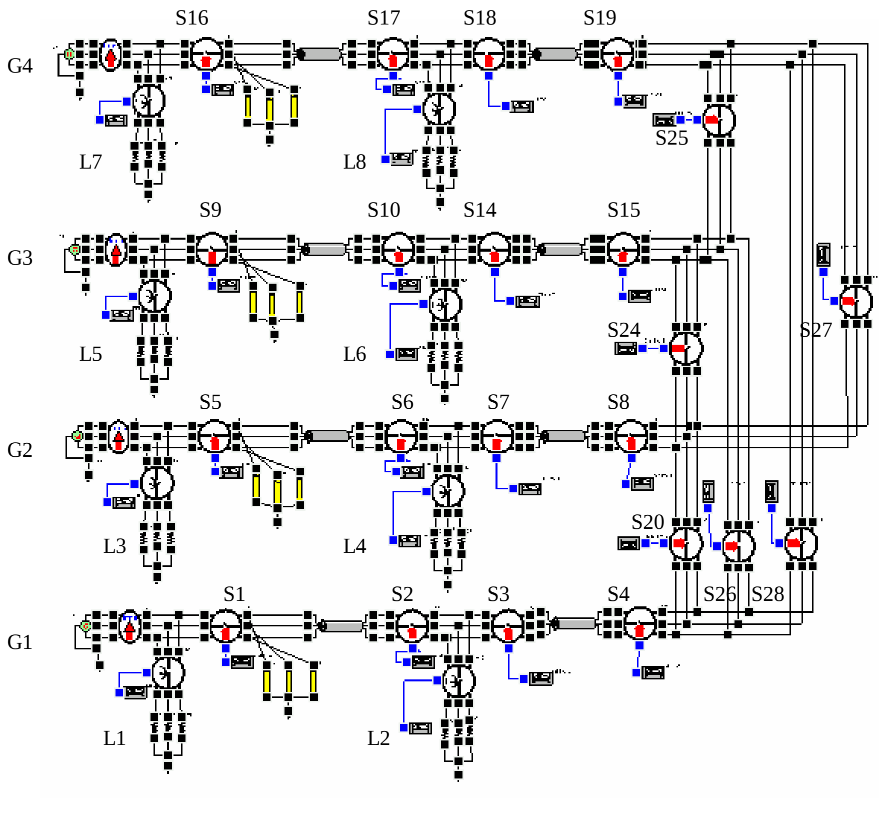


Figure 4.22 VTB Schematic of Distribution Network III

The initialization of the MAS is shown in Appendix A.3.

A. Service Restoration with Selection of Best Path

For this simulation, a fault, F1, as shown in Figure 4.21, was simulated at 0.1 seconds.

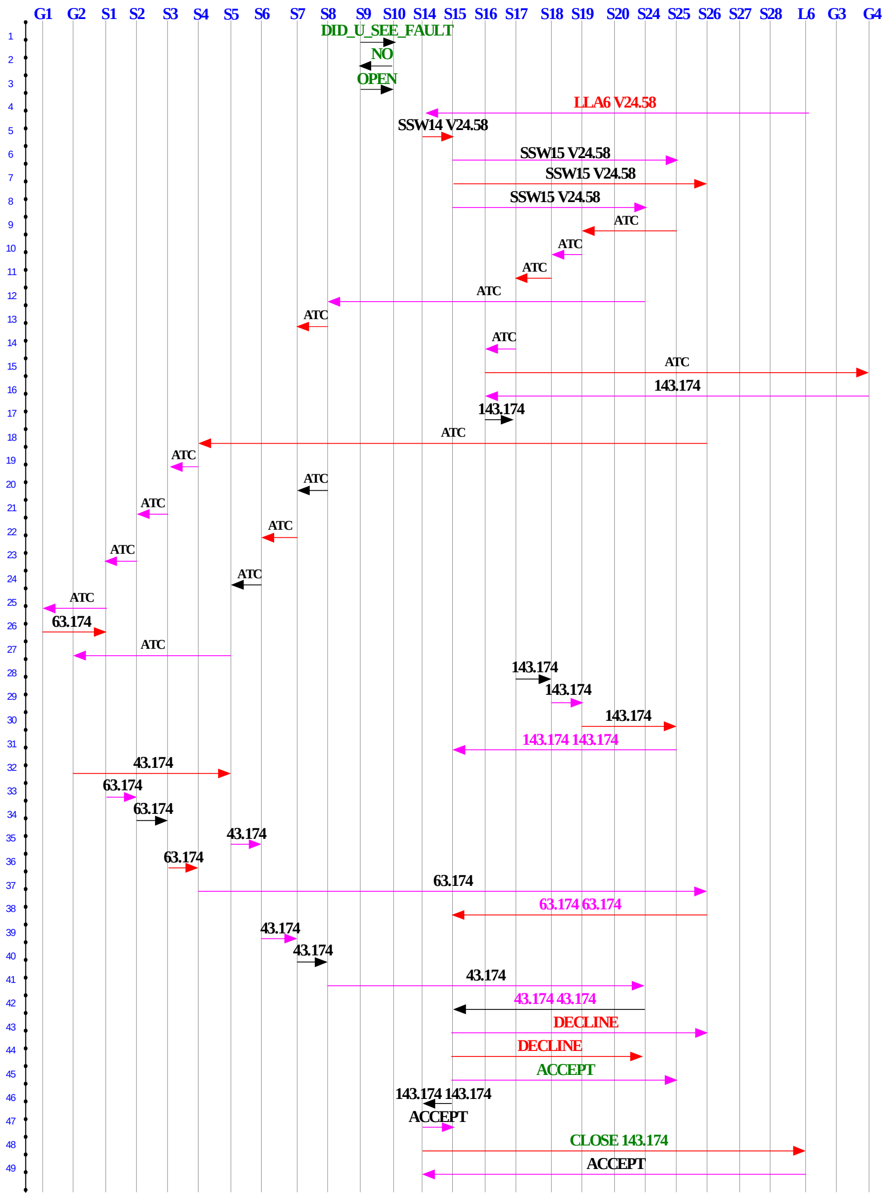


Figure 4.23 Trace of Messages between Agents for Service Restoration with Best Path

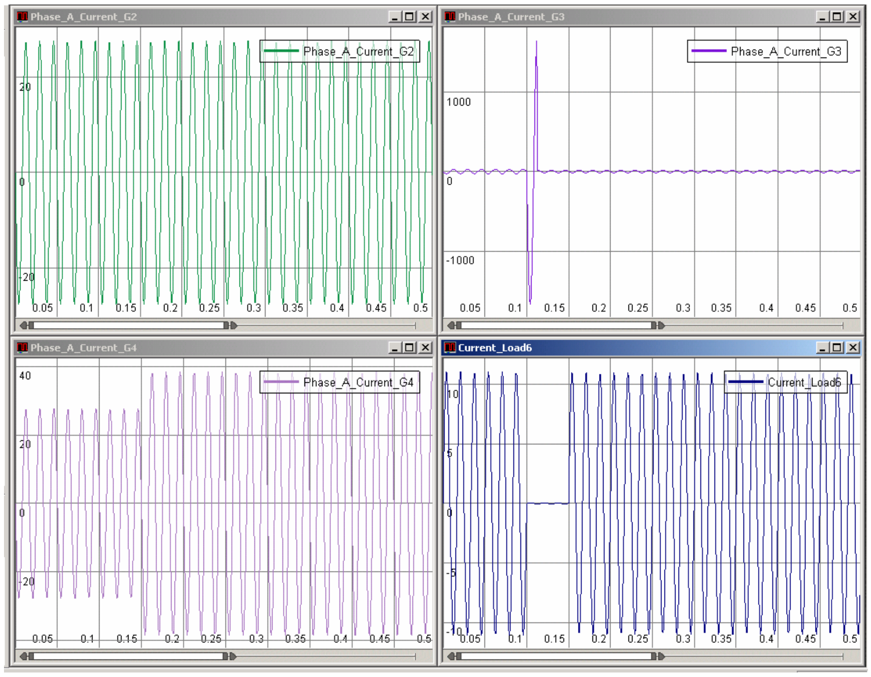


Figure 4.24 Source 2, Source 4, Source 3 and Load 6 Phase A Currents (starting with upper left corner and reading column-wise)

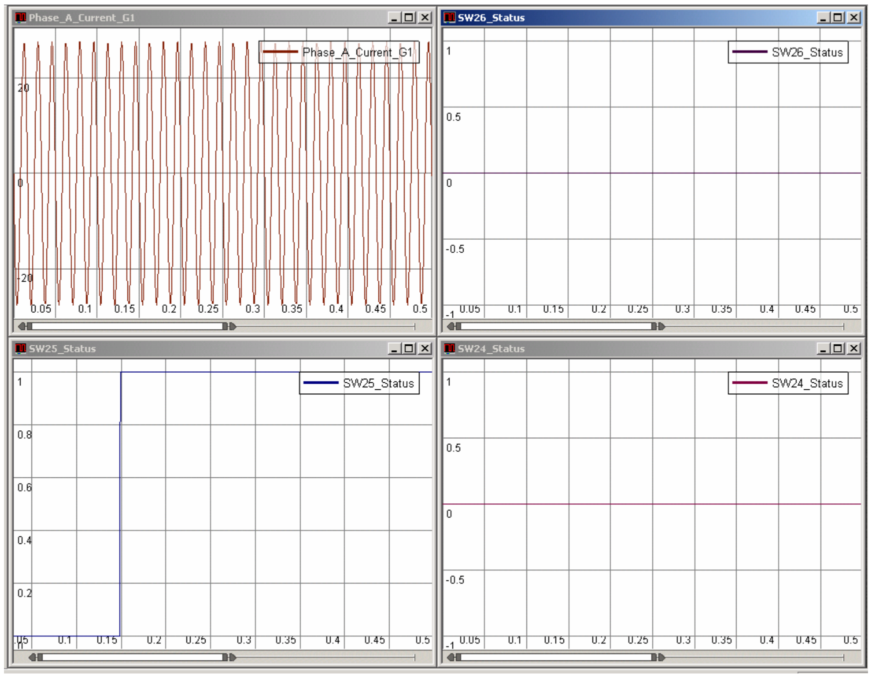


Figure 4.25 Source 1 Phase A Current, Switch 25, Switch 26 and Switch 24 Status (starting with upper left corner and reading column-wise)

The capacities of Sources 1, 2, 3 and 4 are 120A, 100A, 120A and 200A respectively. After the isolation of the fault by opening Switches 9 and 10, Load 6 is de-energized. S15 receives “PROPOSE” and “AVAILATC” from S24-S26, as seen in Messages 42, 31 and 38 respectively in Figure 4.23. S15 accepts S25, as the remaining capacity of Source 4 is the best and declines S24 and S26.

Figure 4.24 shows that Load 6 is restored in 0.037 seconds and the current of Source 4 is increased from 27A to 38A. Figure 4.25 shows that the status of Switch 25 is set to close, but that Switches 24 and 26 remain unchanged.

4.6 Complexity of MAS Development

Developing a MAS is a complex task in itself, because it involves several agents with behaviors, which are recursively called. Figure 4.26 shows the complexity of the developed MAS for a distribution network by UML® class diagram. This class diagram includes main package restoration and three kinds of agents, whose classes are extended from the Agent class of JADE. The Simple Behavior class of JADE was extended to realize the behaviors of different agents.

JADE-VTB communication required the installation of several software packages, as shown in Chapter 3. These software packages required setting the class path and environment variables. The development environment for Java, i.e. J2SDK, Java Web Start for connections between computer, and internet to launch and manage applications of the web, and visual studio.net to change the web service processing directive in the active server pages (ASP), are all needed. The net web service project needed coordination to achieve the communication. Even after all the required communication medium was achieved, the web service failed to respond. Restarting the IIS, after new security and permission settings were applied, solved this problem.

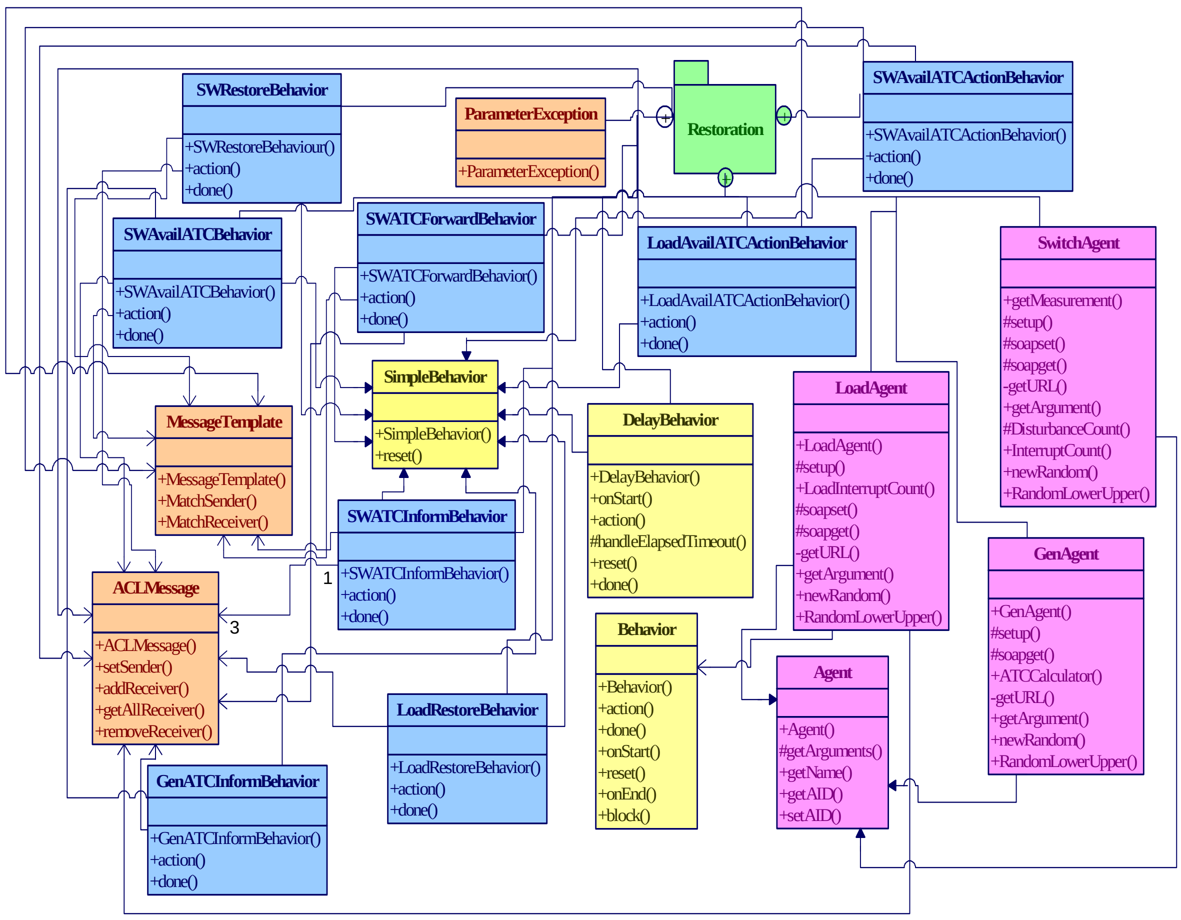


Figure 4.26 UML Class Diagram of Developed Multi-Agent System

The receivers for the message need to be defined by the “addReceiver” method. It was undesirable that the message was sent to the desired receiver and other receivers added previously. JADE does not allow the user to view or print the added receivers. This limitation was overcome by the “removeReceiver” method used every time after sending the message. Since agent messaging is asynchronous, the messages are not received in the order they are sent. Thus, the agent is made to wait for any action unless all replies are received.

After successfully restoring with MAS for a small system, large systems with more than 30 agents were tested. The execution of the MAS failed via the command line as well as by the batch file. The reason for the failure is attributed to the command line becoming too long, as the number of agents, and thereby their arguments increased. Windows 2000 supports only 2047 characters in the command line, either used at command prompt or as a batch file. The agents were divided into two containers, and the agents of the sub container were added into the main container, thereby decreasing the command line length.

Surprisingly, the desired solution was not achieved because the server became busy due to heavy traffic, and the measured data was lost. The solution was achieved by setting the ServerListenBacklog to unlimited instead of its default value of 40 and MaxEndPointConnections to unlimited by commands:

```
IIS5.0 ServerListenBacklog
cscript c:\winnt\system32\inetsrv\adminsamples\adsutil.vbs
SET w3svc/1/ServerListenBacklog 999999
IIS 5.0:
MaxEndPointConnections
cscript c:\winnt\system32\inetsrv\adminsamples\adsutil.vbs SET
w3svc/1/MaxEndpointConnections –1
```

4.7 Conclusions

This chapter investigates a MAS-based solution to achieve an intelligent distribution system able to reconfigure and restore power systems quickly. A MAS is a result of collaboration among several agents. This chapter proposes a MAS with three different types of agents to provide the decentralized solution to distribution system restoration and fill the gaps of a fast and real-time response scheme. The flow diagram of each type of agent is presented along with the details of algorithm. The agents do exchange information and determine a feasible restoration path. The developed architecture is represented by the UML® class diagram. This chapter reveals that a MAS is generalized and does not change with changes in system topology. A successful trace of messages shown in this chapter gives a visualization of the communication between agents. The implementation of the agents has been demonstrated using three different test cases. The MAS features were demonstrated for full restoration, partial restoration with load shedding, and alternative best path selection in case of multiple paths for restoration. The results indicate the MAS can provide a distributed solution for restoration strategies.

CHAPTER V

RESTORATION WITH DISTRIBUTED GENERATORS

This chapter presents restoration with a Distributed Generator (DG) and considers balancing the island by load shedding. Also, it presents preliminary work on the autonomous control of a DG.

5.1 Introduction

Agents have several types and have certain desirable characteristics and modules, as described in previous chapters. Modularity is one of the characteristics of agents and hence the generator agent, load agent and switch agent in Chapter 4 could be reused by the addition of some behaviors discussed in this chapter. Most of the DGs are small capacity generators, and hence they should be placed closer to the customer/load. DGs increase survivability by providing a continuous power supply to important loads and decrease losses by reducing the power supply from the utility/source. If fault isolation does not create an island, the restoration strategies remain the same as developed in Chapter 4. The restoration strategy needs an additional load shedding scheme if the fault isolation results in islanding. Islanding is not a fault scenario, so all switches in the island are originally closed, and the DG starts overloading in order to supply all of the loads in the island. Hence quick control actions are necessary to coordinate and shed the unnecessary loads. However, in an islanding scenario, the maximum unbalance in the island can be 50% in the worst case [60]. To mitigate this unbalance, agent mobility is also necessary apart from its communicative, reactive and collaborative properties. Mobile agents can move within the same platform from container to container. Also, this chapter develops methods of control where DGs compensate for the load fluctuation of the distribution system.

5.2 Changes in Restoration Strategy with the Introduction of DG

The scheme developed in Chapter 4 was the Radial Restoration Scheme (RRS); this chapter will focus on developing Radial Restoration Scheme with DG (RRSDG). The differences in the two schemes are illustrated in Figure 5.1. To strengthen the discussion above, Figure 5.2 demonstrates an example system with several fault scenarios.

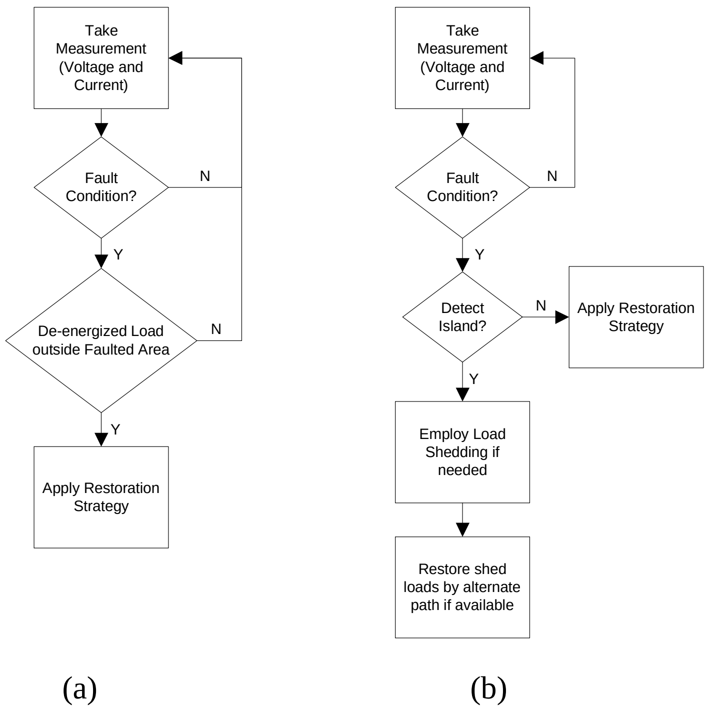

Take Measurement (Voltage and Current)
Fault Condition?
N
Y
De-energized Load outside Faulted Area
N
Y
Apply Restoration Strategy
(a)
Take Measurement (Voltage and Current)
Fault Condition?
N
Y
Detect Island?
N
Apply Restoration Strategy
Y
Employ Load Shedding if needed
Restore shed loads by alternate path if available
(b)

Figure 5.1 Flow Chart for (a) RRS and (b) RRSDG Schemes

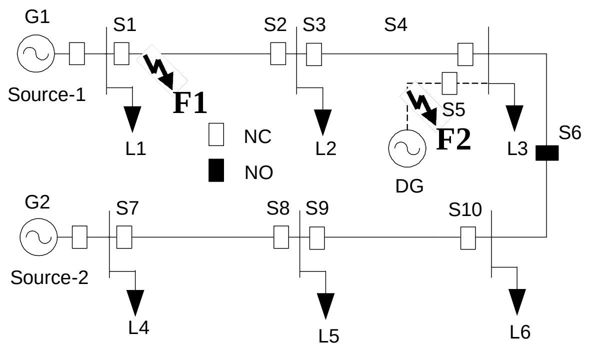


Figure 5.2 Single Line Diagram of Two-Generator System

A. Fault at F1

After the isolation of Fault F1, as shown in Figure 5.2, the normally open S6 will close, depending on the available restoration capacity in the RRS scheme. Some of the loads may not be restored if the available capacity is less than the total loads.

With the RRSDG scheme, a special islanding scenario has occurred. The DG will try to ramp up and may overload if the loading is greater than its capacity. Therefore, to protect the DG and maintain power balance in the island, shedding the extra loads as soon as possible is essential. After shedding the loads, the island is carefully maintained, however, there may be some shed loads. The normally open S6 can be closed, depending on available restoration capacity to supply the shed loads.

B. Fault at F2

After the isolation of fault at F2, the utility or main generator will supply the loads that were supplied by the DG; no switching action is necessary in the RRSDG scheme.

5.3 Model Development

Islanding studies were done in MATLAB using the demo file "power_turbine.mdl" to understand the dynamics of the generator when DG control acts during islanding, step increase of load, and/or step decrease of load. Based on the analysis, a DG model was built using a PI controller to the level of details required for the restoration, because the constant voltage source model of VTB is not enough to simulate these types of dynamics. Also, this model is necessary for the application of MAS to restoration of distribution system with multiple DGs. A hierarchical entity for breaker, which opens to create and island, was also built.

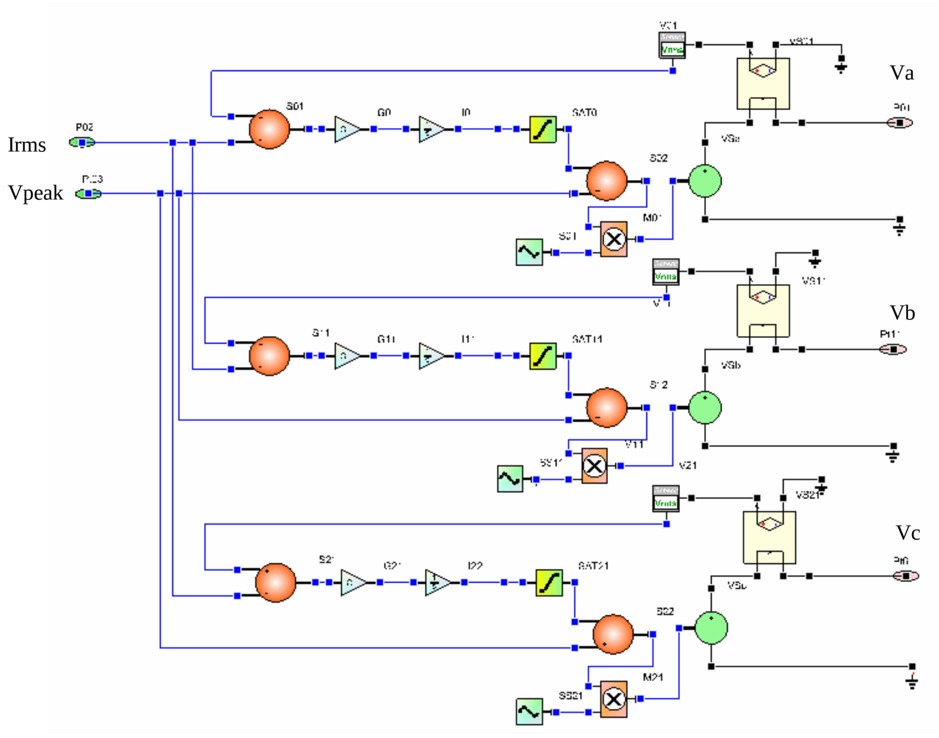
5.3.1 Distributed Generator Model and Validation

As Figure 5.3 (a) shows, this model employs a PI type controller to maintain the constant voltage mode until it reaches its current limit. After the current limit is reached, the generator works in constant current mode, keeping the current at the limiting value. Figure 5.3 (b) shows the icon in VTB for the developed model.

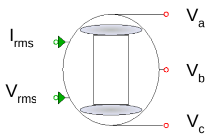
The one line diagram of the test system shown in Figure 5.4 (a), is used to validate the developed model. Figure 5.4 (b) shows the VTB schematic of the test system. The DG capacity is 2 p.u., and the rating of Load 1 is 0.5 p.u.

Case 1

At t = 2.5 seconds, Load 1 rises from 0.5 p.u. to 6 p.u., thus the DG is overloaded by 4 p.u.. The voltage drops from 1 p.u. to 0.3 p.u. as the DG now operates in constant current mode, as shown in Figure 5.5 (a).

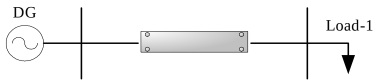


(a)

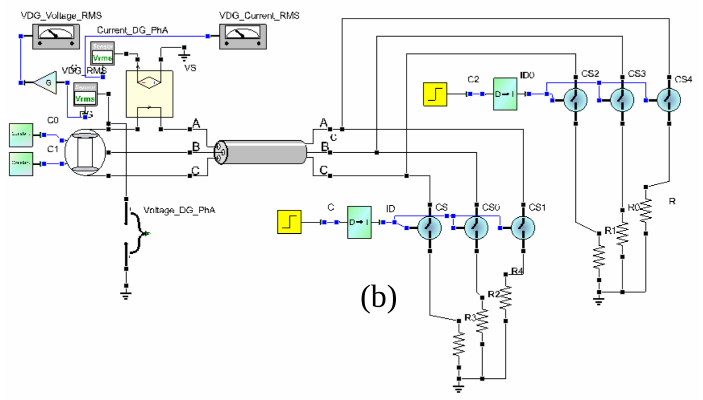


(b)

Figure 5.3 (a) DG Model (b) DG Icon



(a)

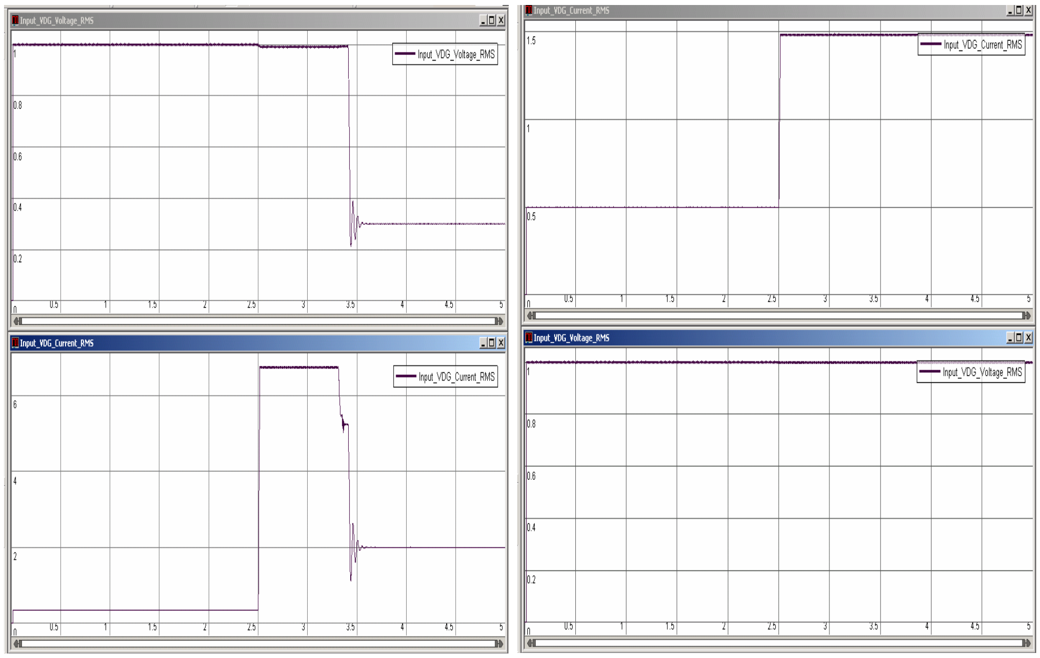


(b)

Figure 5.4 (a) Test System for DG Model (b) VTB Schematic of Test System for DG Model

Case 2

At t = 2.5 seconds, Load 1 rises from 0.5 p.u. to 1.5 p.u., and the DG is within its capacity of 2 p.u.. The increase in load does not violate the DG's capacity. Thus, as shown in Figure 5.5b, the DG supplies the load current instantaneously and the voltage remains constant at 1 p.u.



(a) (b)

Figure 5.5 DG Terminal Voltage and Current (a) Case 1 (b) Case 2

5.3.2 *Breaker Hierarchical Entity*

The Breaker Agents (BAs) obtain current by using the CCVS block and real power is obtained by voltage measurement, delay, integrator, sum, multiplier and gain blocks in VTB. Through the RPC, agents access this current and real power. In Figure 5.6, the RPC identity is set as 61-62 for current measurement and real power. This real power is then used to find the direction of current through the breaker. If more than one

BA is present in the system, then the RPC ID can be increased to 71-72, 81-82 and so forth.

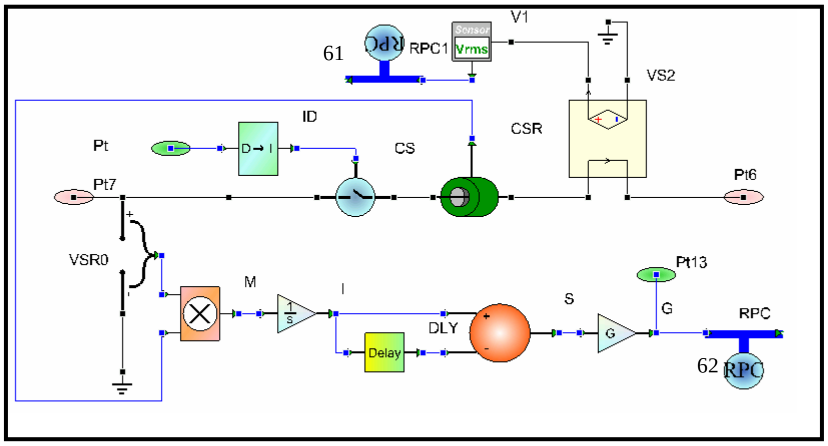


Figure 5.6 Hierarchical Model of Three-Phase Current Measurement with Icon

5.4 Agents

The Breaker Agent (BA) and the Move Agent (MA) were introduced in the MAS to do restoration of distribution systems with DGs.

5.4.1 Breaker Agent

The BA is initialized as:

BA (ID, Status, NumMoveNeighbors, ID_MoveNeighbors)

where,

ID: Agent Identification Number

Status: Open/Close

NumMoveNeighbors: Number of moving agent neighbors

ID_MoveNeighbors: Agent Identification Number of moving agent neighbors

The BA informs the Move Agent (MA) of the current through its terminal when it opens up. In a power system with multiple DGs, the DGs can feed back to the utility in light load conditions. The BA also informs the MA about the real power flow through the breaker, which is later used to find the direction of the current.

5.4.2 Move Agent

The MA is initialized as:

MA (ID, NumGenNeighbors, ID_GenNeighbors, NumLoadNeighbors, ID_LoadNeighbors, NumBrkNeighbors, ID_BrkNeighbors)

where,

ID: Agent Identification Number

NumGenNeighbors: Number of generator neighbors

ID_GenNeighbors: Agent Identification Number of generator neighbors

NumLoadNeighbors: Number of load neighbors

ID_LoadNeighbors: Agent Identification Number of load neighbors

NumBrkNeighbors: Number of breaker neighbors

ID_BrkNeighbors: Agent Identification Number of breaker neighbors

Load Agents (LAs) and Generator Agents (GAs) inform the MA about the current loading and current generation respectively. When the MA receives a message from BA, it checks to see if the loads need to be shed. BA then informs the loads to be shed.

5.5 Agent Development

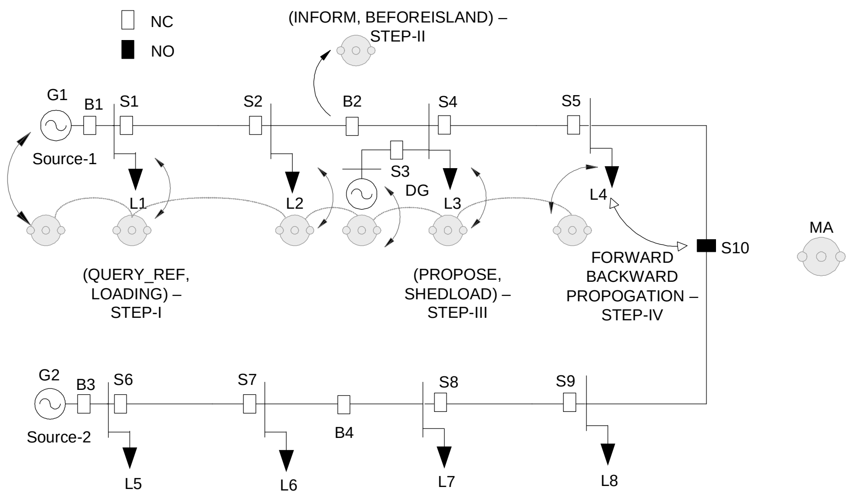
Chapter 4 developed behaviors for SA, LA and GA. For load shedding under an islanded condition, behaviors were added to GA and LA. Table 5.1 shows the short description of behaviors added to LA and GA, along with behaviors developed for BA and MA. A detailed description of behaviors responsible for load shedding under islanded conditions and associated with each agent is given next.

Table 5.1. Description of Behaviors associated with Agents for Restoration with DG

Agents	Behaviors	Receive Message	Send Message/Signal	Description
Breaker Agent	IslandRestore Start	-	- performative "INFORM" and conversation ID "BEFOREISLAND"	Sends current magnitude and real power through the breaker to MA.
Move Agent	MoveLoad Request	-	- performative "QUERY_REF" and conversation ID "LOADING"	Requests current loading from loads and current generation from generators when it reaches the respective locations while it moves.
	MoveLoad Receive-Loading	- performative "INFORM" and conversation ID "LOADING"	-	Receives current loading from loads and current generation from generators when it reaches the respective locations while it moves.
	IslandDetect	- performative "INFORM" and conversation ID "BEFOREISLAND"	- performative "PROPOSE" and conversation ID "SHEDLOAD"	Receives current magnitude and real power from BA. Decides the load that has to be shed and sends OPEN to LA
Load Agent	MoveLoad Reply	- performative "QUERY_REF" and conversation ID "LOADING"	- performative "INFORM" and conversation ID "LOADING"	Receives the query from MA and sends its current loading.
	MoveLoad Action	- performative "PROPOSE" and conversation ID "SHEDLOAD"	- Send OPEN/CLOSE signals to itself	Sends OPEN signal to itself when content of the received message is OPEN
Generator Agent	MoveGen Reply	- performative "QUERY_REF" and conversation ID "LOADING"	- performative "INFORM" and conversation ID "LOADING"	Receives the query from MA and sends its current generation and maximum capacity.

5.5.1 Breaker Agent

1. IslandRestoreBehaviour: This behavior corresponds to Step-II in Figure 5.7. On its opening the breaker agent informs MA with performative “INFORM” and Conversation ID “BEFOREISLAND” when it is encountered during the mobility of MA. The contents of the message are the current flowing through the breaker and the real power flowing through the breaker.



NC
NO
(INFORM, BEFOREISLAND) – STEP-II
G1
B1
S1
S2
B2
S4
S5
Source-1
S3
DG
L1
L2
L3
L4
MA
S10
FORWARD BACKWARD PROPOGATION – STEP-IV
(QUERY_REF, LOADING) – STEP-I
(PROPOSE, SHEDLOAD) – STEP-III
G2
B3
S6
S7
S8
S9
Source-2
B4
L5
L6
L7
L8

Figure 5.7 Messages Flow for Load Shedding between Different Agents

5.5.2 Move Agent

1. MoveLoadRequestBehaviour: This behavior corresponds to Step-I in Figure 5.7. The agent sends a message with performative “QUERY_REF” and ConversationID “LOADING” to LA neighbors and GA neighbors encountered during its mobility.
2. MoveLoadReceiveLoadingBehaviour: This behavior corresponds to Step-I in Figure 5.7. The agent receives a message with performative “INFORM” and

ConversationID "LOADING" from its LA neighbors and GA neighbors encountered during its mobility. The MA extracts and saves the message received from GA, categorizing the content as local name, associated switch ID, maximum capacity and current generation. The MA extracts and saves the message received from LA, categorizing the content as local name, associated switch ID, status and current loading. Name denotes the unique agent identification; status refers to critical/vital or non-critical/non-vital; associated switch ID refers to the immediate downstream switch neighbor.

3. MoveLoadIslandDetectBehavior: This behavior corresponds to Steps-II and III in Figure 5.7. The agent receives a message with performative "INFORM" and ConversationID "BEFOREISLAND" from BA encountered during its mobility. The message contains the agent local name, real power and current magnitude through the breaker. The magnitude of current through the breaker indicates the supply from utility/source to downstream loads. If the current direction is upstream, then no loads need to be shed. If the current direction is downstream, then the magnitude of loads to be shed is at least equal to the current magnitude through the breaker. The local name sent by BA and the associated switch ID sent by the GA and the LA help in identifying the GAs and LAs downstream of BA. These GAs and LAs form the island. The MA has information from each of the downstream GAs about the pre-island generation and maximum capacity, and pre-island loading of downstream LAs, as described in (2) above. The available generation is obtained by subtracting the pre-island generation from the maximum

capacity. The amount of load to be shed is thus determined by subtracting the available generation from the current magnitude sent by BA (utility was supplying before disconnection). The MA then sends performative "PROPOSE" and ConversationID "SHEDLOAD" to the LA encountered first during its mobility and continues until the load shed is at least equal to the calculated amount of load to be shed.

5.5.3 Load Agent

1. MoveLoadReplyBehavior: This behavior corresponds to Step-I in Figure 5.7. The agent receives a message from MA with performative "QUERY_REF" and ConversationID "LOADING" when it is encountered during the mobility of MA. The agent, in reply, sends a message with content local name, associated downstream switch ID, status and current loading with performative "INFORM" and ConversationID "LOADING" to MA.
2. MoveLoadActionBehavior: This behavior corresponds to Step-III in Figure 5.7. The agent receives a message from MA with performative "PROPOSE" and ConversationID "SHEDLOAD" when it is encountered during the mobility of MA. If the message content is open/close, then the LA sends an open/close signal to itself.

5.5.4 Generator Agent

1. MoveGenReplyBehavior: This behavior corresponds to Step-I in Figure 5.7. The agent receives a message from MA with performative "QUERY_REF" and

ConversationID “LOADING” when it is encountered during the mobility of MA. The agent in reply sends a message with content local name, associated downstream switch ID, maximum capacity and current generation with performative “INFORM” and ConversationID “LOADING” to MA.

5.6 Results of Load Shedding and Restoration

A power system test case was simulated in VTB. Chapter 4 describes the development of a MAS for restoration of distribution systems without DG. This section focuses on restoration of distribution systems with DG under islanded conditions using JADE to develop the multi-agent framework. The agents only communicate with neighboring agents. Communication messages for agents participating in restoration and load shedding only are shown in the results. Mobile agent moves from container to container to perform the desired task. The MAS is tested on a system with DG for shedding loads after breaker opening leading to separation from the utility. The high priority of critical/vital loads must be considered during load shedding and restoration through alternate path. For simplicity, one-line diagrams of the test cases do not show the load breakers. The per unit data in the test cases is obtained on a MVA base of 0.1 MVA and kV base of 6.6 kV.

5.6.1 Load shedding without Restoration

This section considers two systems; Test System 1, which is a 6-node system, and Test System 2, which is an 18-node modified ship system. One case is considered for Test System 1 and two cases are considered for Test System 2.

I *Test System I*

The system in Figure 5.8 is a 6-node system with one generator, one DG and four loads; Loads 2 and 3 are critical/vital, and Loads 1 and 4 are non-critical/non-vital. Tables 5.2 and 5.3 show the bus data and branch data in per unit for the system.

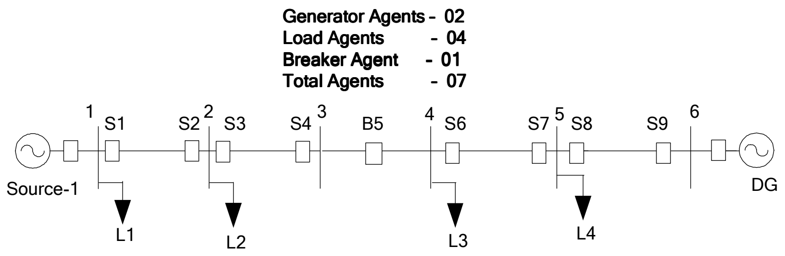


Figure 5.8 Distribution Network I

Table 5.2. 6-Node Test System – Bus Data

Bus No.	Voltage (p.u.)	Max Generator Capacity (p.u.)	Load (p.u.)
1	1.0	Slack	10
2	-	-	10
3	-	-	0
4	-	-	15
5	-	-	20
6	1.0	20	0

Table 5.3. 6-Node Test System – Branch Data

From Bus	To Bus	Branch Resistance (p.u.)
1	2	0.001
2	3	0.0001
3	4	0.0001
4	5	0.0001
5	6	0.001

In the simulation, breaker B5 opens at 0.348 seconds. Figure 5.9 shows the VTB schematic. The initialization of the MAS is shown in Appendix A.4.

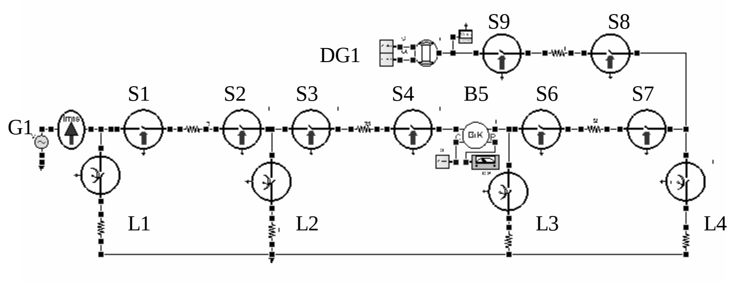


Figure 5.9 VTB Schematic of Distribution Network I

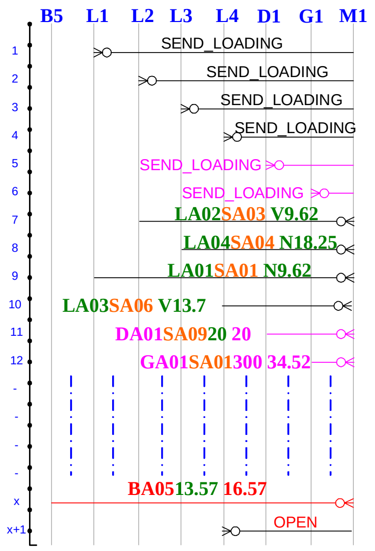


Figure 5.10 Trace of Messages between Agents for Load Shedding

When the system is in a normal state, the MA gets the information about current loading and current generation from loads and generators as seen by Messages 1-12 in Figure 5.10. This receiving of loading and generation information occurs until the system operates in normal state. After the opening of the breaker, a power imbalance occurs in the island by 12 p. u. This opening of the breaker leads to asynchronous Message x with performative “INFORM” and conversation ID “BEFOREISLAND” from BA5 (Breaker Agent 5)-MA1 (Move Agent 1). The MA sheds the non-vital load L4 shown in Message x+1. Figure 5.11 shows that Load 4 is shed in 0.032 seconds after the opening of the breaker, incorporating no delay for physical constraints.

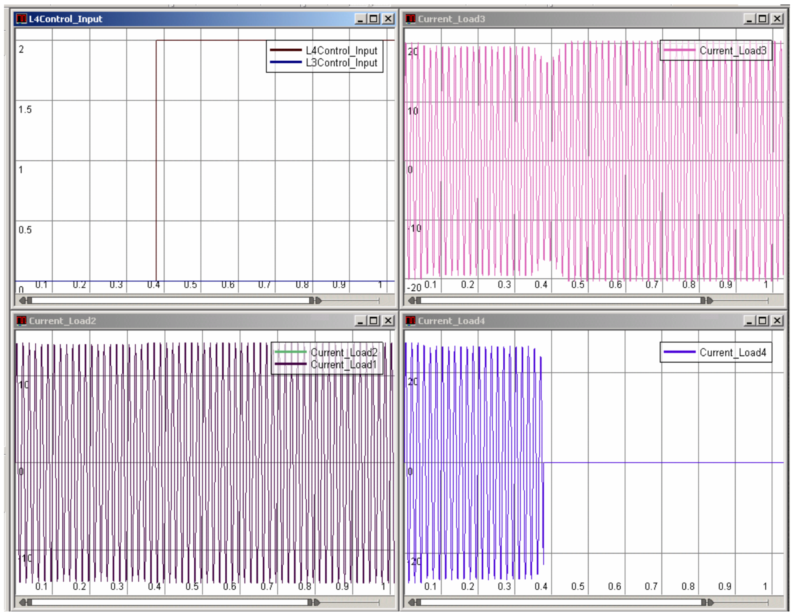


Figure 5.11 Load 4 and Load 3 Control Signals, Load 1 and Load 2 Currents, Load 3 Current, Load 4 Current (starting with upper left corner and reading column wise)

The current of DG1 decreases from 20 p.u. to 14.4 p.u. and the current of Generator 1 decreases from 34.5 p.u. to 19.6 p.u. Figure 5.12 shows voltages and currents of DG and the generator.

II Test System 2

The system in Figure 5.13 is the Healy icebreaker ship system, whose data was obtained from [61] and modified to consider only real power loads. The modified ship system is an 18-node system with one generator, three DGs and six loads; Loads 3 and 6 are critical/vital, and Loads 1, 2, 4 and 5 are non-critical/non-vital. Tables 5.4 and 5.5 show the bus data and branch data in per unit for the system.

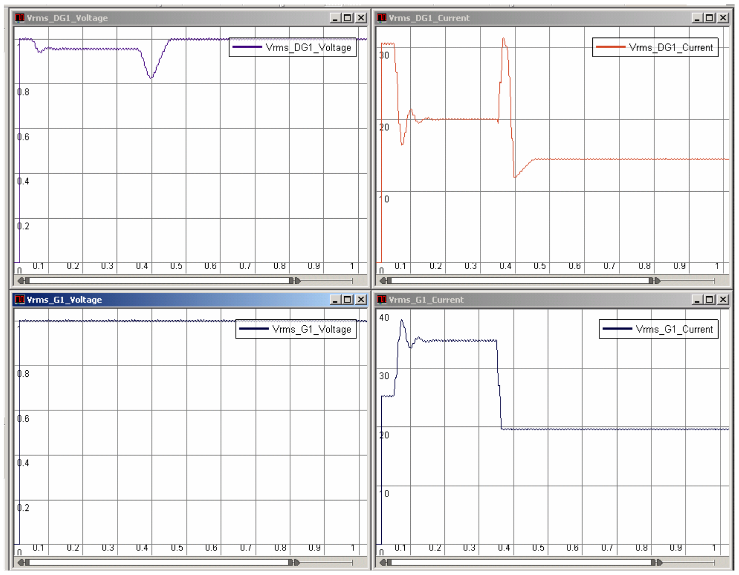


Figure 5.12 DG1 and G1 RMS Voltages, DG1 and G1 Currents (starting with upper left corner and reading column wise)

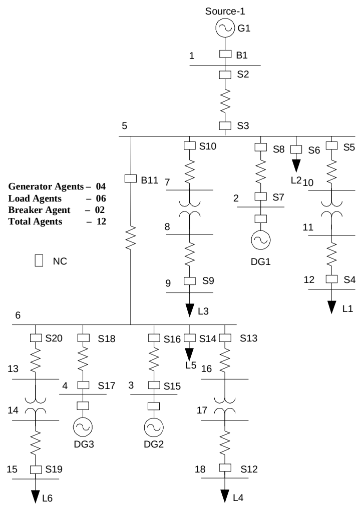


Figure 5.13 Distribution Network II (Modified Ship System)

Table 5.4. 18-Node Modified Ship System – Bus Data

Bus No.	Voltage (p.u.)	Max Generator Capacity (p.u.)	Load (p.u.)
1	1.02	Slack	0
2	1.02	20.5	0
3	1.02	20.2	0
4	1.02	20.13	0
5	-	-	1.4
6	-	-	1.27
7	-	-	0
8	-	-	0
9	-	-	19.36
10	-	-	0
11	-	-	0
12	-	-	19.06
13	-	-	0
14	-	-	0
15	-	-	19.36
16	-	-	0
17	-	-	0
18	-	-	18.93

Table 5.5. 18-Node Modified Ship System – Branch Data

From Bus	To Bus	Branch Resistance (p.u.)	Transformer Tap
1	5	1.6344E-6	-
2	5	1.4770E-3	-
3	6	1.5268E-3	-
4	6	1.5855E-3	-
5	6	0.6464E-6	-
5	7	1.6834E-6	-
5	10	2.4369E-6	-
6	13	2.8090E-6	-
6	16	3.3765E-6	-
7	8	2.0125E-6	1.0
8	9	2.3196E-6	-
10	11	2.0125E-6	1.0
11	12	2.3169E-6	-
13	14	0.20125E-3	1.0
14	15	2.6816E-6	-
16	17	0.20125E-3	1.0
17	18	2.8579E-6	-

A. Case 1

In the simulation, Breaker B1 opens at 0.055 seconds. Figure 5.14 shows the VTB schematic. The initialization of the MAS is shown in Appendix A.5.

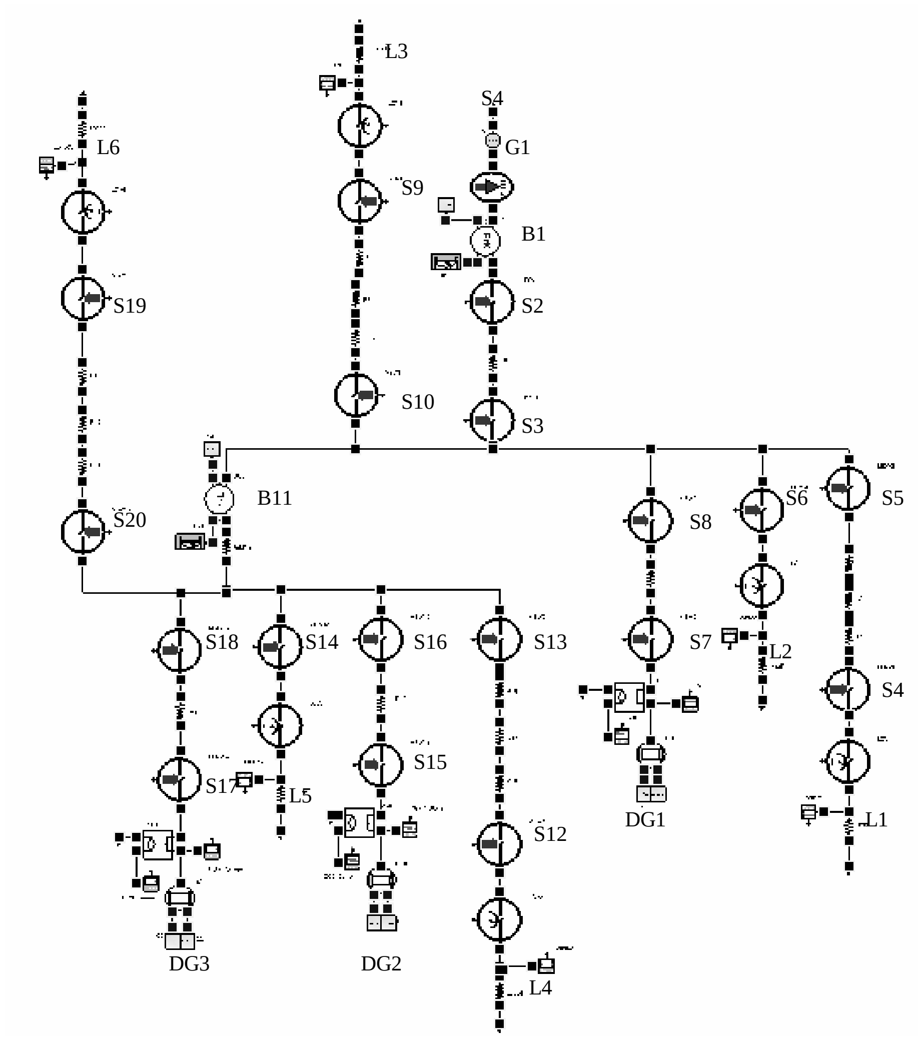


Figure 5.14 VTB Schematic of Distribution Network – II (Modified Ship System)

When the system is in a normal state, the MA gets the information about current loading and current generation from loads and generators, as seen from Messages 1-20 in

Figure 5.15. After the opening of the breaker, a power imbalance occurs in the island by 25.3 p.u. This opening of the breaker leads to asynchronous Message x with performative

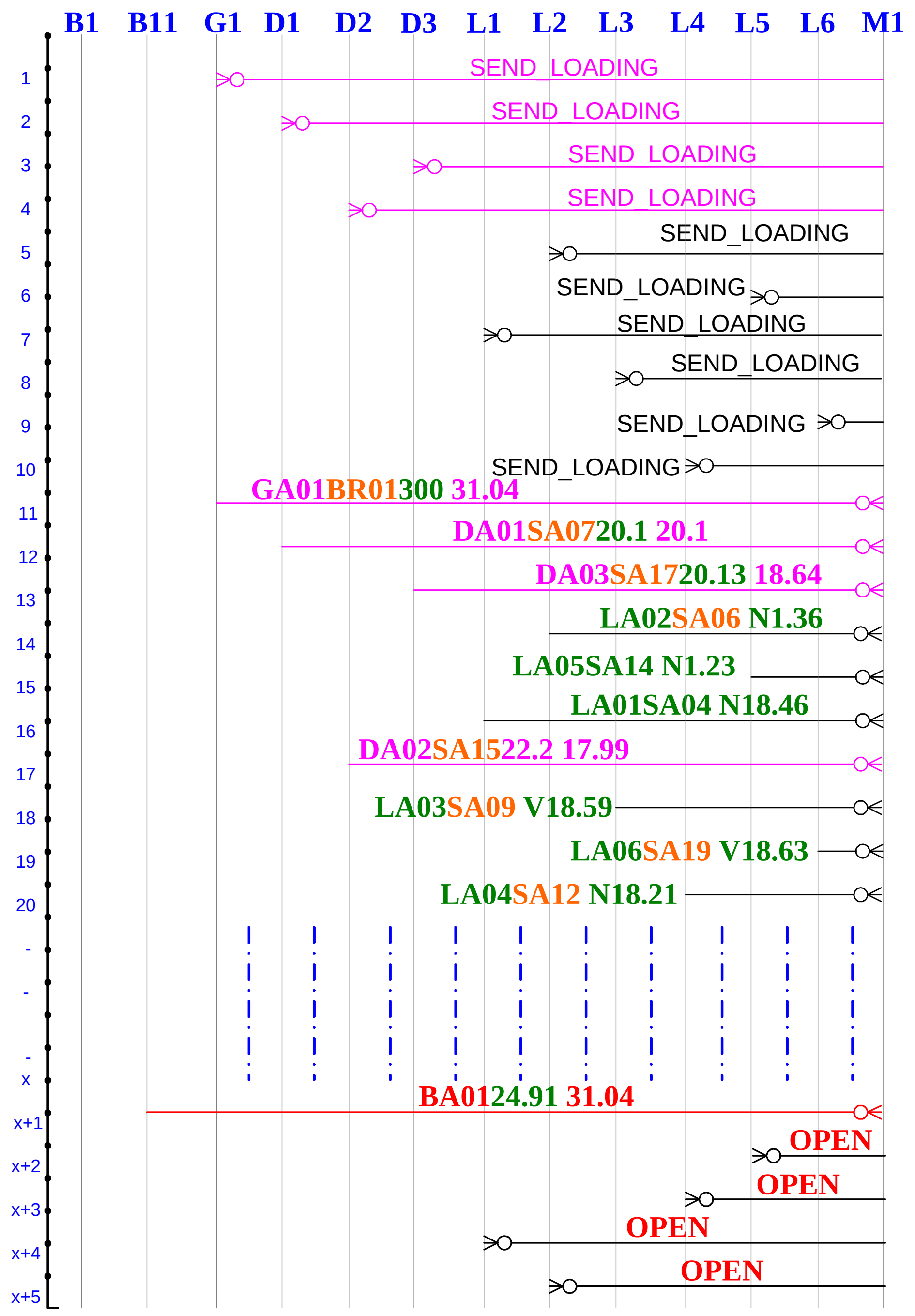


Figure 5.15 Trace of Messages between Agents for Load Shedding – Modified Ship System (Case 1)

"INFORM" and conversation ID "BEFOREISLAND" from BA2 (Breaker Agent 2)-MA1 (Move Agent 1). The MA sheds the non-vital loads L1, L2, L4 and L5, as shown in Messages x+1 to x+5. Figure 5.16 shows that Loads 1, 2, 4 and 5 are shed in a maximum of 0.006 seconds after the opening of the breaker, incorporating no delay for physical constraints. The current of DG1 decreases from 20 p.u. to 14.5 p.u., the current of DG2 decreases from 18 p.u. to 11.2 p.u., current of DG3 decreases from 18 p.u. to 11.65 p.u. and current of Generator 1 decreases from 31.05 p.u. to 0 p.u.. Figure 5.17 shows the voltages and currents of DGs and generator.

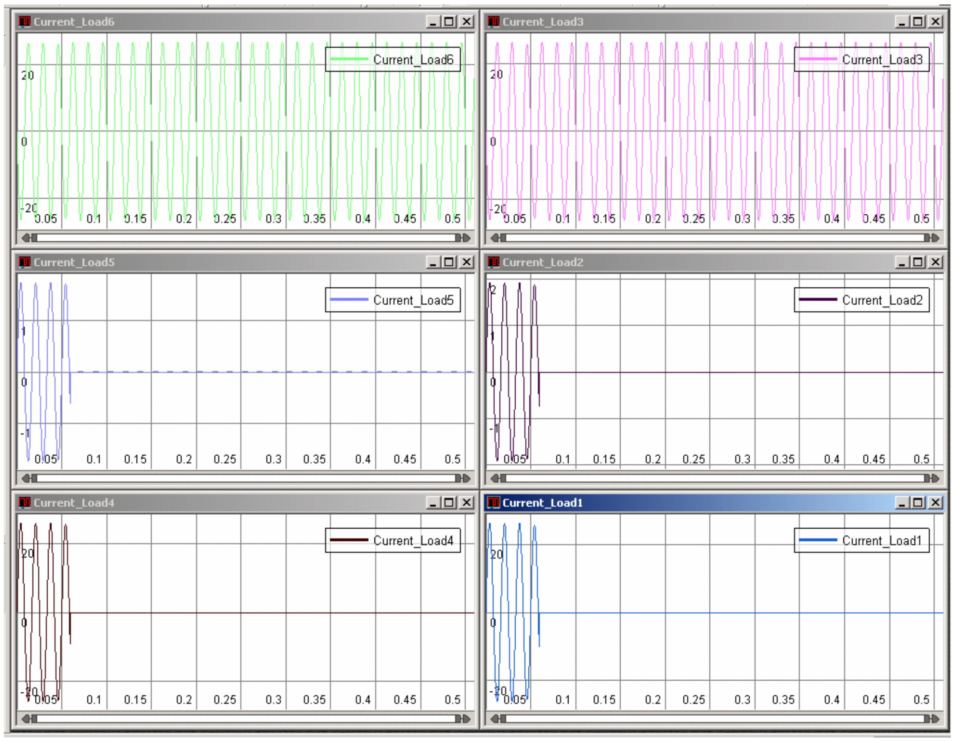


Figure 5.16 Load 6, 5, 4, 3, 2 and 1Currents (starting with upper left corner and reading column wise) – Modified Ship System (Case 1)

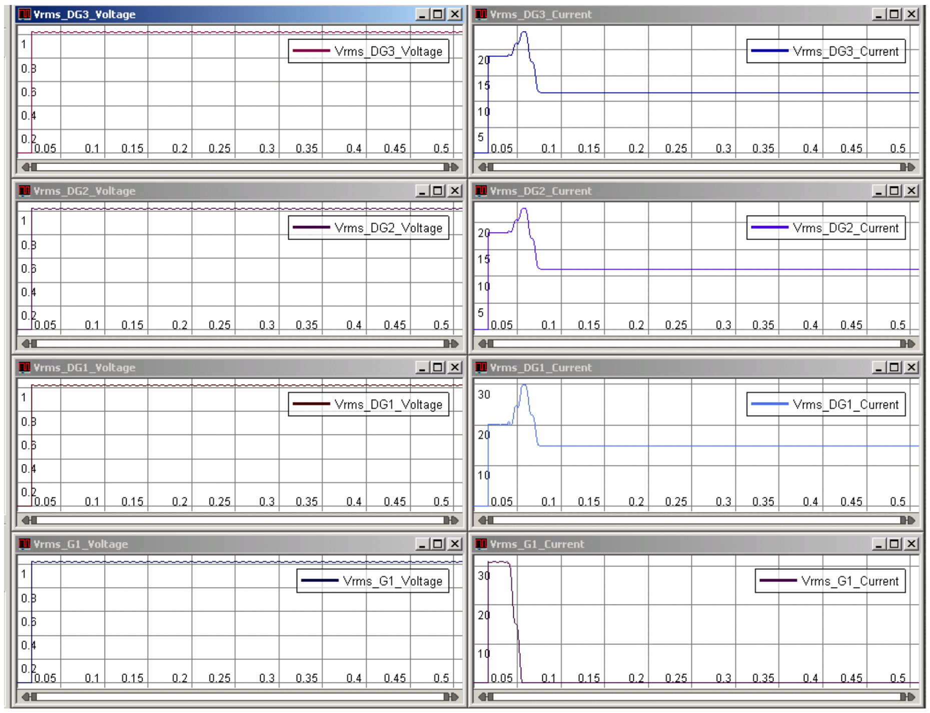


Figure 5.17 DG3, DG2, DG1 and G1 RMS Voltages, DG3, DG2, DG1 and G1 Currents (starting with upper left corner and reading column wise) – Modified Ship System (Case 1)

B. Case 2

Figures 5.13 and 5.14 show the system and its schematic. In the simulation, breaker B11 opens at 0.183 seconds. Only Load 5 was increased from 1.27 to 15 p.u.; Tables 5.4 and 5.5 show the rest of the data. The initialization of the agents is similar to case 1 above.

When the system is in a normal state, the MA gets the information about present loading and current generation from loads and generators, as seen by Messages 1-20 in Figure 5.18. After the opening of the breaker, a power imbalance occurs in the island by 10.97 p.u. This opening of the breaker leads to asynchronous Message x with performative “INFORM” and conversation ID “BEFOREISLAND” from BA11 (Breaker

Agent 11)-MA1 (Move Agent 1). The MA sheds the non-vital load L5 as shown in Message x+1. Figure 5.19 shows that Load 5 is shed in 0.004 seconds after opening of breaker, incorporating no delay for physical constraints. The current of DG1 decreases from 21.45 p.u. to 20.5 p.u., current of DG2 decreases from 22.2 p.u. to 18.2 p.u., current of DG3 decreases from 23 p.u. to 18.8 p.u. and current of Generator 1 decreases from 34.7 p.u. to 19.6 p.u. Figure 5.20 shows the voltages and currents of DGs and generator.

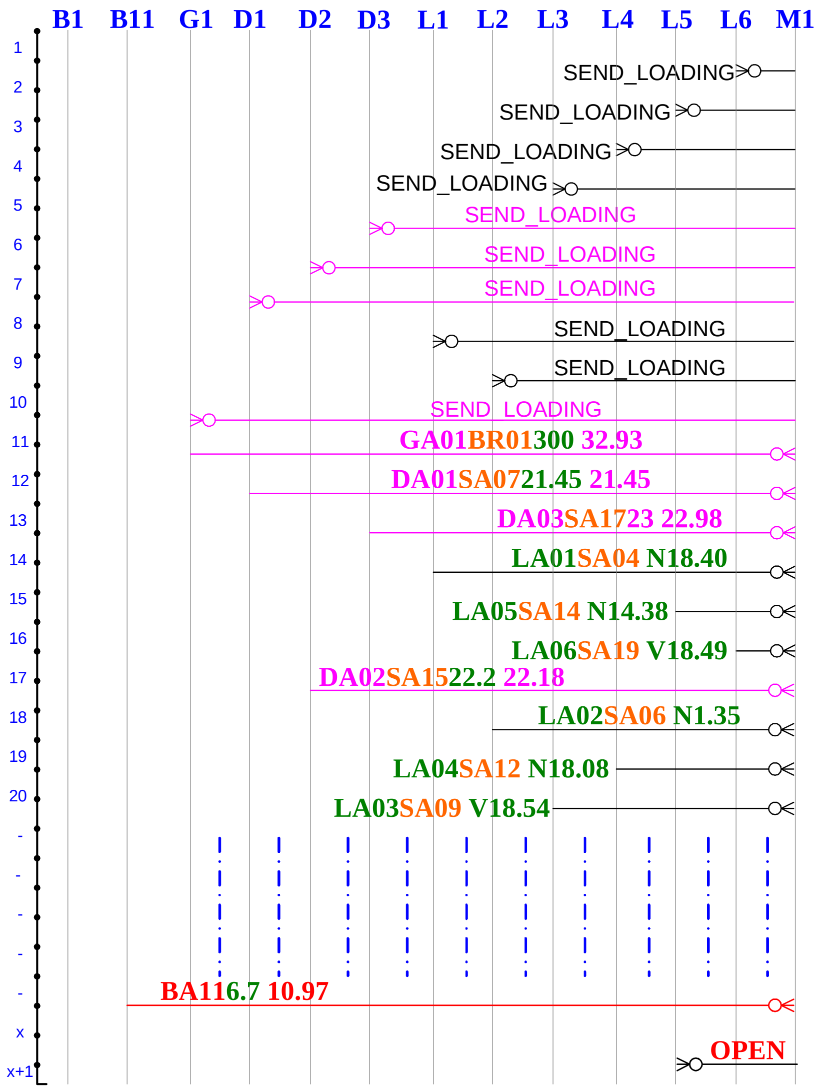


Figure 5.18 Trace of Messages between Agents for Load Shedding – Modified Ship System (Case 2)

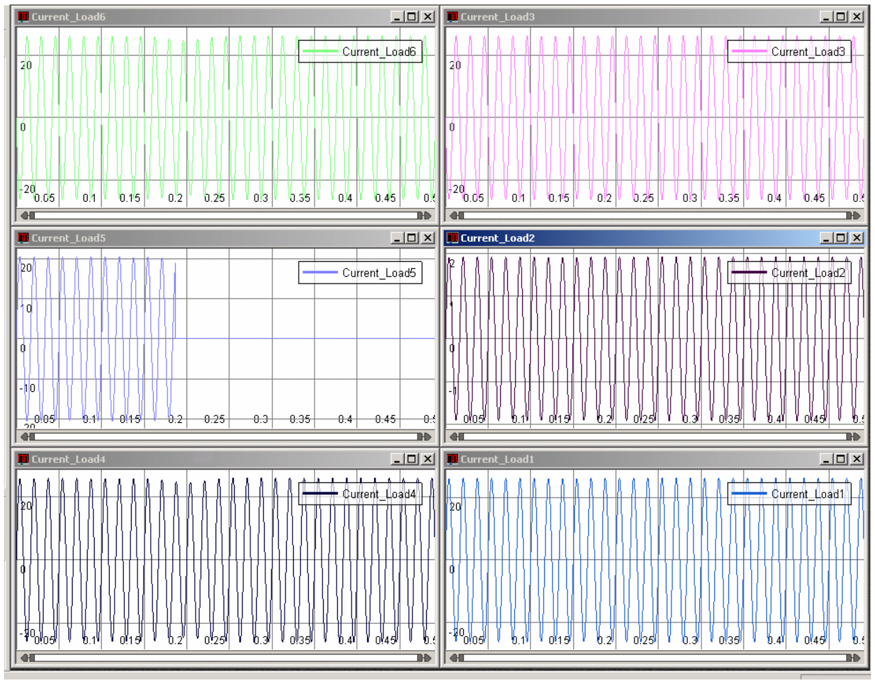


Figure 5.19 Load 6, 5, 4, 3, 2 and 1 Currents (starting with upper left corner and reading column wise) – Modified Ship System (Case 2)

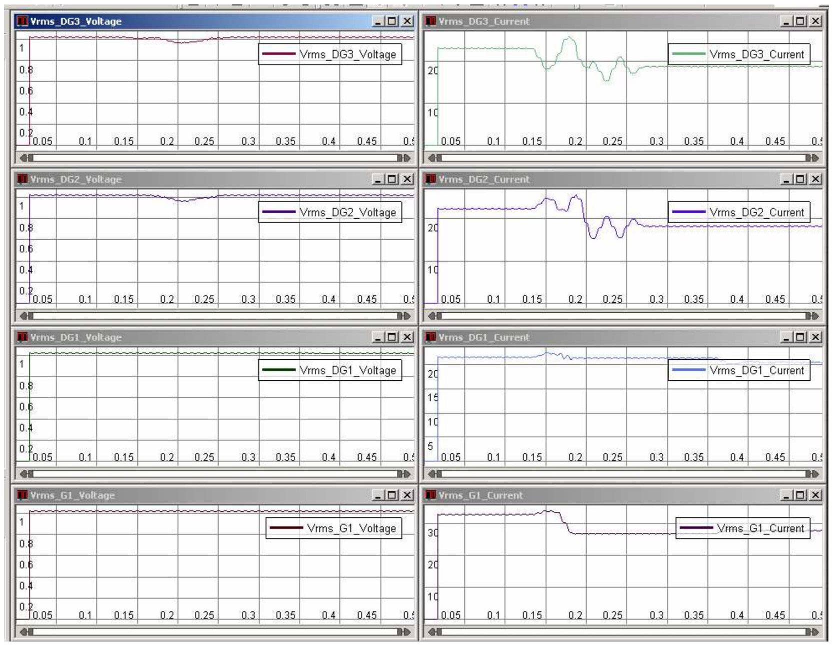


Figure 5.20 DG3, DG2, DG1 and G1 RMS Voltages, DG3, DG2, DG1 and G1 Currents (starting with upper left corner and reading column wise) – Modified Ship System (Case 2)

5.6.2 *Load shedding with Restoration*

This section considers two systems: Test System 1, an 8-node system and Test System 2, a 21-node modified ship system. One case is considered for Test System 1 and Test System 2 each.

I *Test System I*

The system in Figure 5.21 is an 8-node system with two generators, one DG and six loads; Loads 2, 4 and 5 are critical/vital, and Loads 1, 3 and 6 are non-critical/non-vital. Tables 5.6 and 5.7 show the bus data and branch data in per unit for the system.

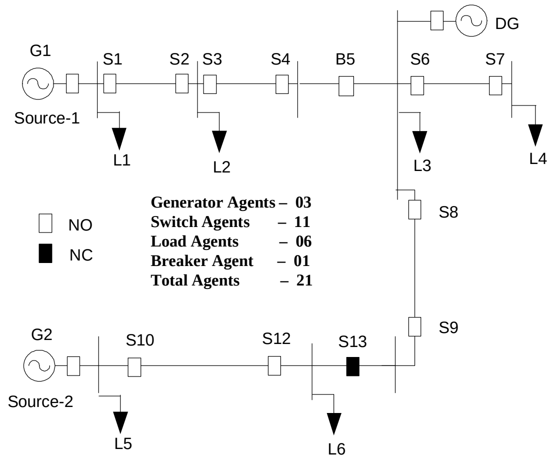


Figure 5.21 Distribution Network III

In the simulation, breaker B5 opens at 0.237 seconds. Figure 5.22 shows the VTB schematic. The initialization of the MAS is shown in Appendix A.6.

When the system is in a normal state, the MA gets the information about present loading and current generation from loads and generators, as seen by Messages 1-12 in Figure 5.23.

Table 5.6. 8-Node Test System – Bus Data

Bus No.	Voltage (p.u.)	Max Generator Capacity (p.u.)	Load (p.u.)
1	1.0	Slack	25
2	-	-	25
3	-	-	0
4	1.0	25	16
5	-	-	20
6	1.0	Slack	25
7	-	-	25
8	-	-	0

Table 5.7. 8-Node Test System – Branch Data

From Bus	To Bus	Branch Resistance (p.u.)
1	2	1.50E-6
2	3	1.50E-6
3	4	1.58E-6
4	5	1.00E-3
6	7	1.00E-3
7	8	1.50E-6
4	8	1.00E-3

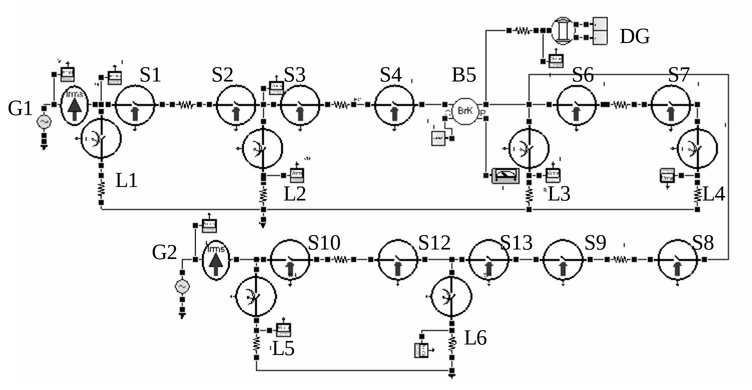


Figure 5.22 VTB Schematic of Distribution Network III

After the opening of the breaker, a power imbalance occurs in the island by 11.75 p.u. This opening of the breaker leads to asynchronous Message x with performative

"INFORM" and conversation ID "BEFOREISLAND" from BA5 (Breaker Agent 5)-MA1 (Move Agent 1). The MA sheds the non-vital load L3 shown in Message x+1.

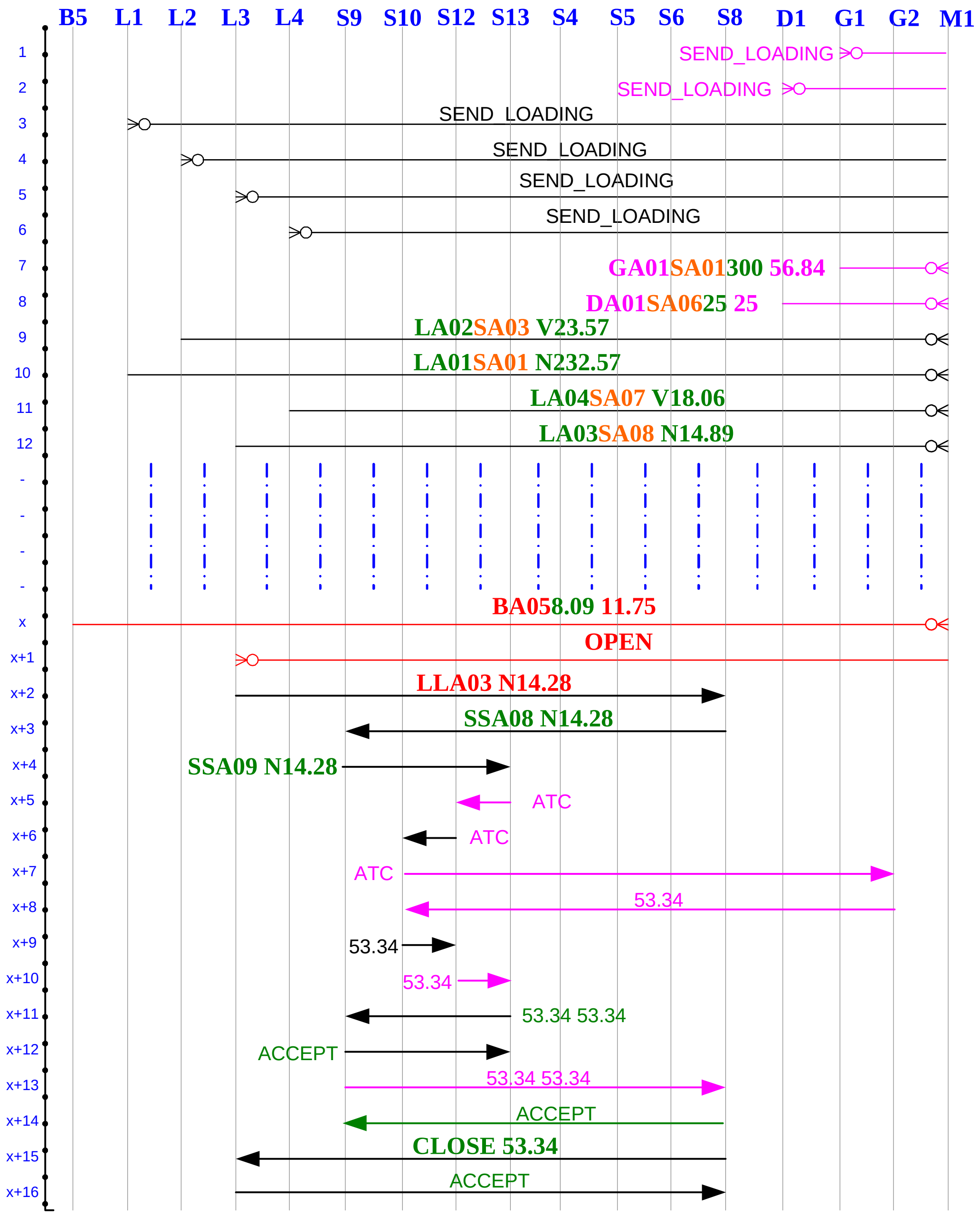


Figure 5.23 Trace of Messages between Agents for Load Shedding and Restoration

Shedding of L3 leads to an asynchronous message with performative “QUERY_REF” and conversation ID “RESTORE” from L3 (Load Agent 2)-S8 (Switch Agent 3) as shown in Message x+2, which is then forwarded through S9 to S13. S13 changes the conversation ID to “ATC” and sends a message to G2 (Generator Agent 2) through S12 and S10 as shown in Messages x+5 to x+7.

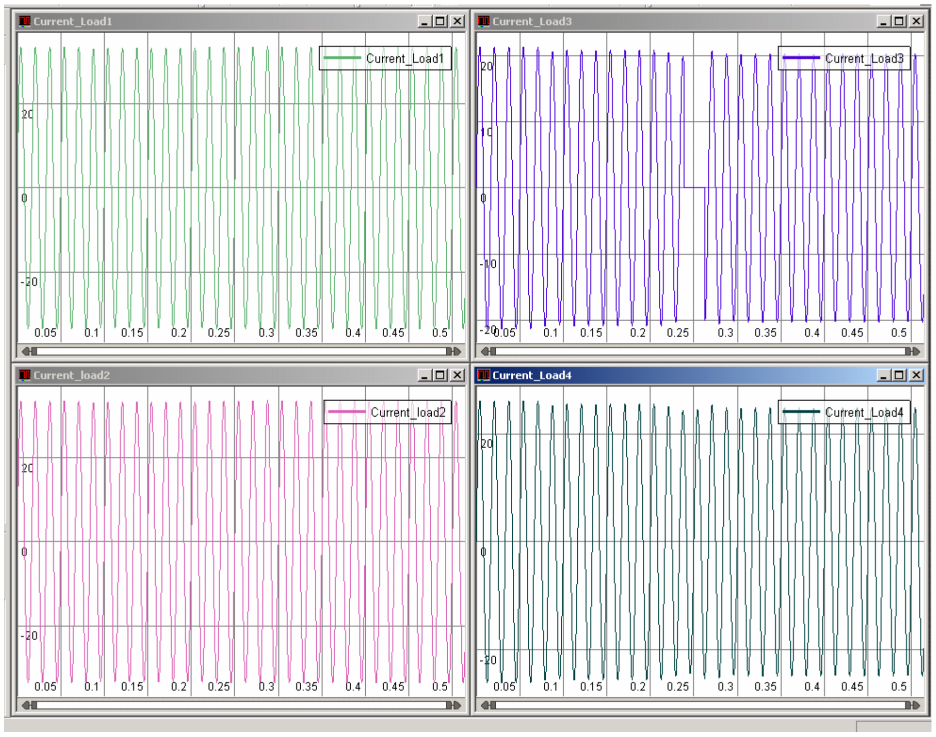


Figure 5.24 Load 1, 2, 3 and 4 Currents (starting with upper left corner and reading column wise)

G2 changes the performative to “INFORM” and passes its remaining capacity to S13 through S10 and S12 as shown in Messages x+8 to x+10. S13, which is normally open, changes the performative to “PROPOSE”, and conversation ID to “AVAILATC” and forwards the message to S8 through S9. S8 sends “close” to L3 as shown in Message x+15, and Load 3 is restored, since the remaining capacity of G2 is enough to supply this load. Figure 5.24 shows that Load 3 is shed in 0.002 seconds after opening of breaker, incorporating no delay for physical constraints. Load 3 is restored in 0.024 seconds after the load is shed. The current of DG1 remains same, and Generator 1 decreases from 57 p.u. to 47.5 p.u.,while the current of Generator 2 increases from 47 p.u. to 54.8 p.u. Figure 5.25 shows the voltages and current of DGs and generators.

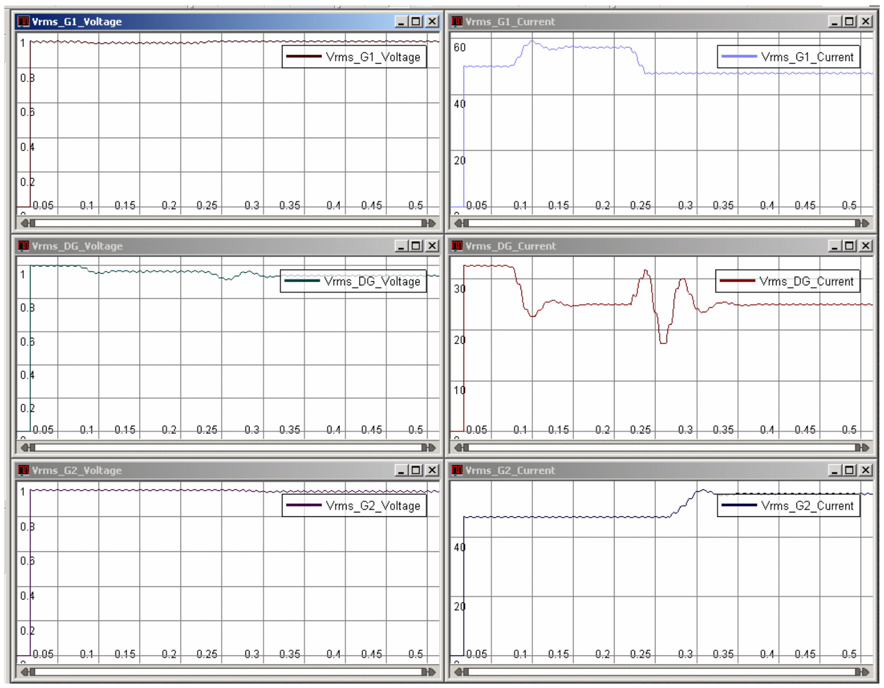


Figure 5.25 G1, DG and G2 RMS Voltages, G1, DG and G2 RMS Currents (starting with upper left corner and reading column wise)

II Test System II

The modified ship system shown in Figure 5.26 is a 21-node system with two generators, two DGs and six Loads; Loads 1, 4, 5 and 6 are critical/vital, and Loads 2 and

3 are non-critical/non-vital. Tables 5.8 and 5.9 show the bus data and branch data in per unit for the system.

In the simulation, breaker B6 opens at 0.0818 seconds. Figure 5.27 shows the VTB schematic. The initialization of the MAS is shown in Appendix A.7.

When the system is in a normal state, the MA gets the information about current loading and current generation from loads and generators as seen by Messages 1-14 in Figure 5.28. After the opening of the breaker, a power imbalance occurs in the island by 14.4 p.u.

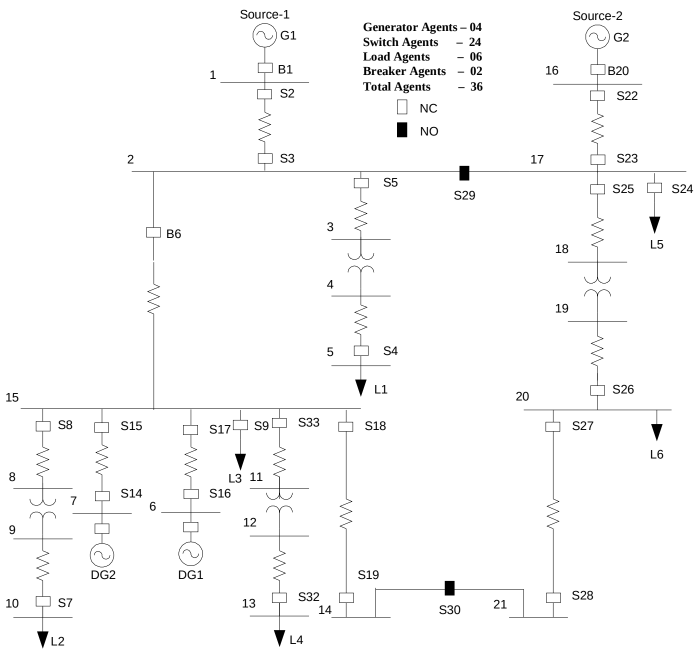


Figure 5.26 Distribution Network – IV (Modified Ship System)

The breaker opening leads to asynchronous Message x with performative “INFORM” and conversation ID “BEFOREISLAND” from BA6 (Breaker Agent 6)-MA1 (Move Agent 1). The MA sheds the non-vital load L3 shown in Message x+1.

Table 5.8. 21-Node Modified Ship System – Bus Data

Bus No.	Voltage (p.u.)	Max Generator Capacity (p.u.)	Load (p.u.)
1	1.02	Slack	0
2	-	-	0
3	-	-	0
4	-	-	0
5	-	-	19.2
6	1.02	22	0
7	1.02	22	0
8	-	-	0
9	-	-	0
10	-	-	19.37
11	-	-	0
12	-	-	0
13	-	-	18.93
14	-	-	0
15	-	-	15
16	1.02	Slack	0
17	-	-	12
18	-	-	0
19	-	-	0
20	-	-	19.06
21	-	-	0

Table 5.9. 21-Node Modified Ship System – Branch Data

From Bus	To Bus	Branch Resistance (p.u.)	Transformer Tap
1	2	1.6344E-6	-
2	15	0.64641E-6	-
2	3	1.6834E-6	-
3	4	2.0125E-6	1.0
4	5	2.3196E-6	-
8	15	2.8090E-6	-
7	9	0.20125E-3	1.0
9	10	2.6816E-6	-
15	14	1.5268E-3	-
6	15	1.5855E-3	-
11	15	3.3765E-6	-
12	11	0.20125E-3	1.0
13	12	2.8579E-6	-
14	15	1.0000E-3	-
16	17	0.16770E-3	-
17	18	2.4369E-6	
19	18	1.000E-6	1.0
20	19	2.3169E-6	
20	21	1.000E-6	

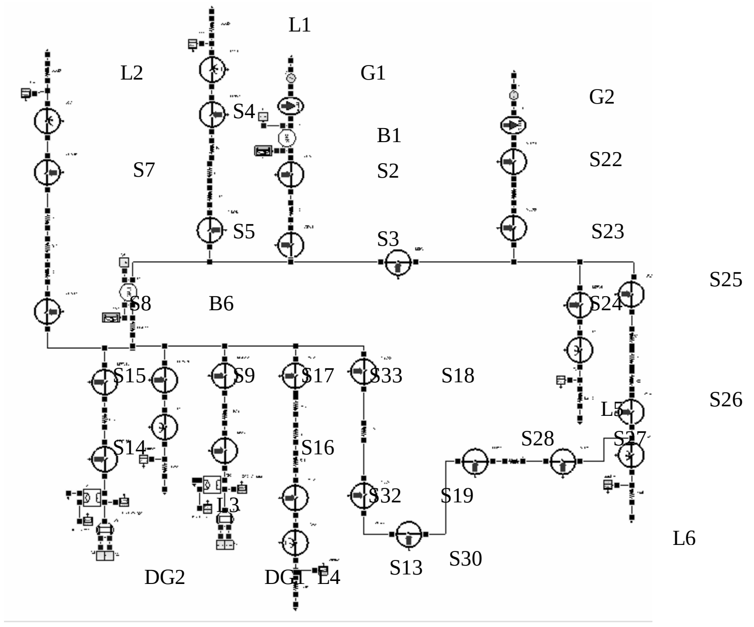


Figure 5.27 VTB Schematic of Distribution Network – IV (Modified Ship System)

Shedding of L3 leads to an asynchronous message with performative “QUERY_REF” and conversation ID “RESTORE” from L3 (Load Agent 2)-S17 (Switch Agent 17) as shown in Message x+2, which is then forwarded to S15, S17 and S18. The message from S15 and S17 are terminated since DA01 and DA02 do not respond to messages with conversation ID “RESTORE”. The message is thus forwarded through S18 and S19 to S30. S30 changes the conversation ID to “ATC” and sends message to G2 (Generator Agent 2) through S28-S22, as shown in Messages x+6 to x+11. G2 changes the performative to “INFORM” and passes its remaining capacity to S30 through S22-S28 as shown in Messages x+12 to x+18. S30, which is normally open, changes the performative to “PROPOSE” and conversation ID to “AVAILATC” and forwards the message to S9 through S18 and S19. S9 sends “close” to L3 as shown in Message x+22, and Load 3 is restored since the remaining capacity of G2 is enough to supply this load.

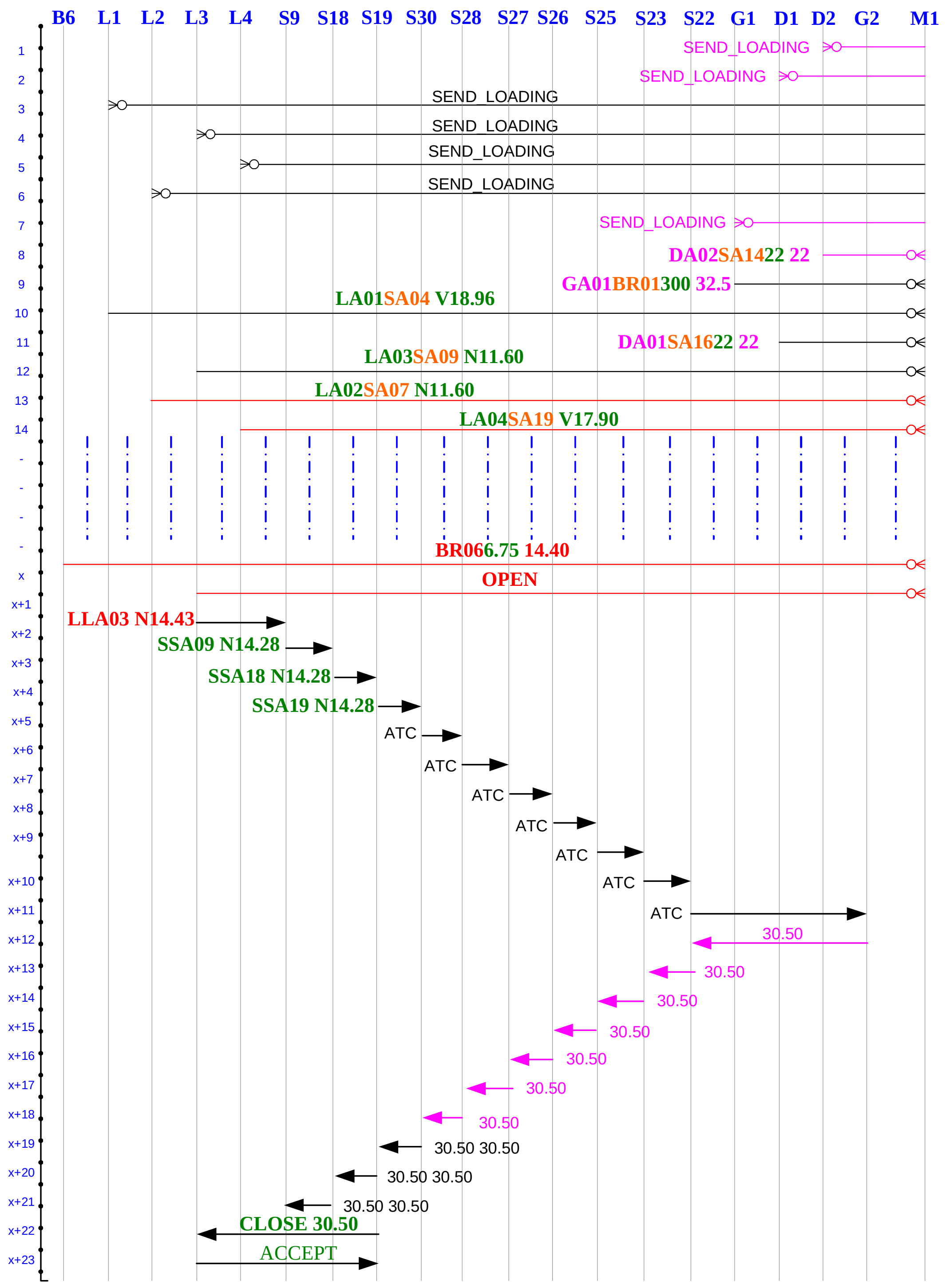


Figure 5.28 Trace of Messages between Agents for Load Shedding and Restoration

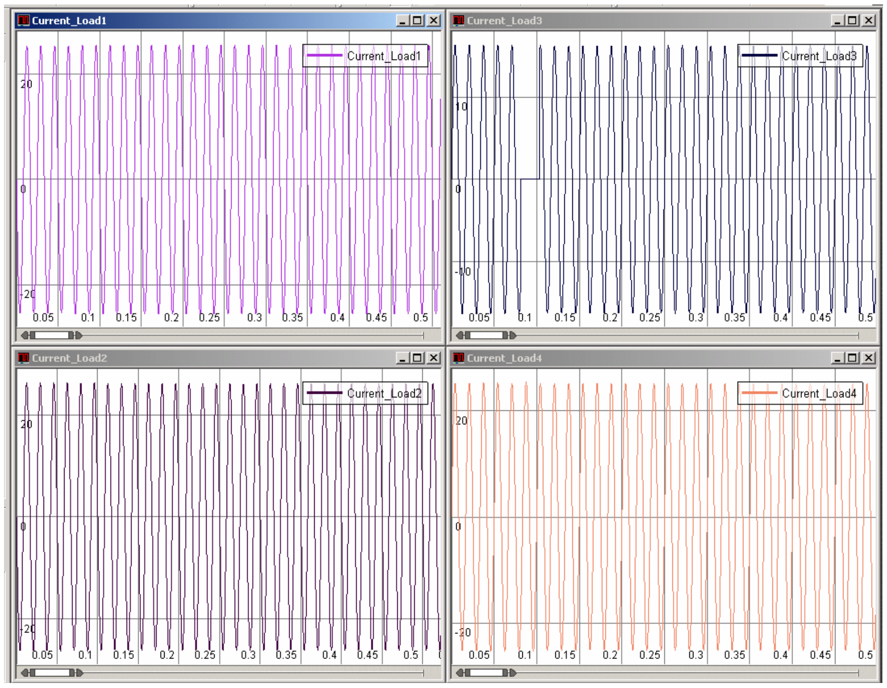


Figure 5.29 Load 1, 2, 3 and 4 Currents (starting with upper left corner and reading column wise)

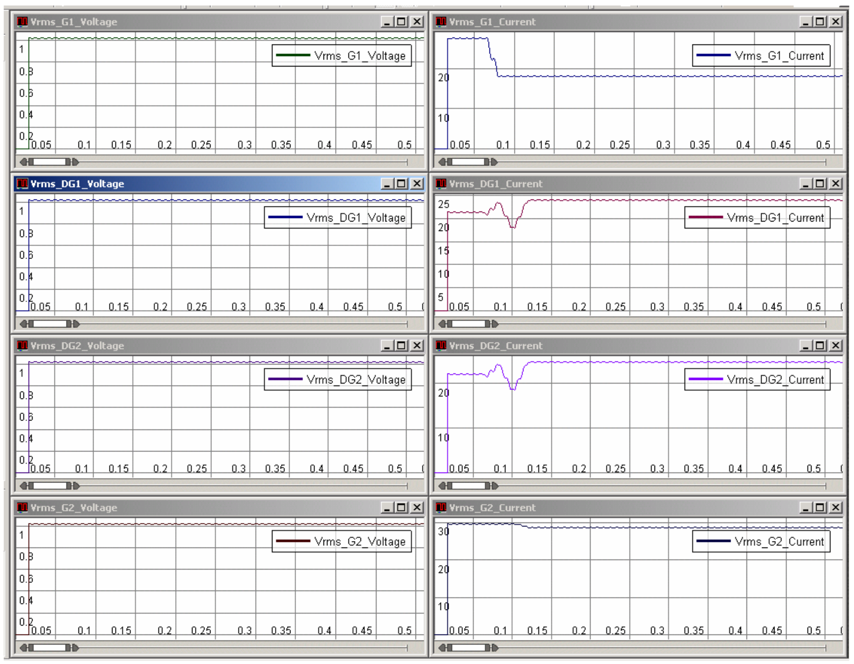


Figure 5.30 G1, DG1, DG2 and G2 RMS Voltages, G1, DG1, DG2 and G2 RMS Currents (starting with upper left corner and reading column wise)

Figure 5.29 shows that Load 3 is shed in 0.006 seconds after the opening of the breaker, incorporating no delay for physical constraints. Load 3 is restored in 0.019 seconds after the load was shed.

The current of DG1 remains the same, Generator 1 decreases from 28.2 p.u. to 18.1 p.u., and the current of Generator 2 increases from 30.8 p.u. to 41.03 p.u. Figure 5.30 shows the voltages and currents of DGs and generators.

5.7 Control of Distributed Generator

In Section 5.6 it was assumed that a DG can adjust to the varying nature of loads under islanded conditions or while performing restoration. This research work proposes a control method to accomplish such fast adjustments of DG output.

5.7.1 Test System

To accomplish such a control, this research considered an IEEE 4-node test system [62]. Figure 5.31 shows the test system with the addition of DG and load variations. The test system was simulated using SIMULINK and the SimPowerSystems library of MATLAB/ SIMULINK. The SIMULINK model of the synchronous generator with the diesel turbine unit and excitation system is used for the DG. Also, the DG is modeled as an independent current source using Equation 5.1. Only real power output is considered for ease of calculation and demonstration of technique.

$$\vec{I}_d = \left(\frac{S_{DG}}{V_d}\right)^* = I_{real} + jI_{reactive} \tag{5.1}$$

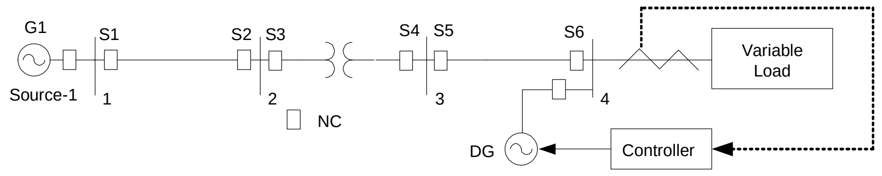


Figure 5.31 Control of DG

5.7.2 Results

First, a steady state solution for the test system was obtained using MATLAB/ SIMULINK. A load flow analysis was also done using radial distribution analysis package (RDAP) [63]. The results obtained from SIMULINK, shown in Table 5.10, were close to that obtained from RDAP, shown in Table 5.11, verifying the model and the steady state solution. The small variation in results is due to difference in line modeling and solution technique. In steady state, the utility delivers less power than the DG, because the DG is modeled as constant current, thus keeping the power constant at 1 MW. The following steps are followed now:

Table 5.10. Voltages and Currents from MATLAB without DG

Bus No	|V| Volts	∠V (Deg)	Branch	|I| Amp	∠I (Deg)	Phase
1	7199.45	0	12	100.41	-31.10	A
1	7199.43	-120	12	129.17	-146.29	B
1	7199.42	120	12	163.68	98.62	C
2	7185.82	-0.07	23p	102.14	-32.67	A
2	7156.8	-120.13	23p	130.64	-147.56	B
2	7162.89	119.68	23p	164.87	97.57	C
3	2392.19	-0.17	23s	304.75	-32.75	A
3	2384.12	-120.26	23s	390.14	-147.58	B
3	2385.35	119.50	23s	492.68	97.56	C
4	2340.46	-0.96	34	304.75	-32.75	A
4	2247.22	-121.74	34	390.14	-147.58	B
4	2270.26	115.75	34	492.68	97.56	C

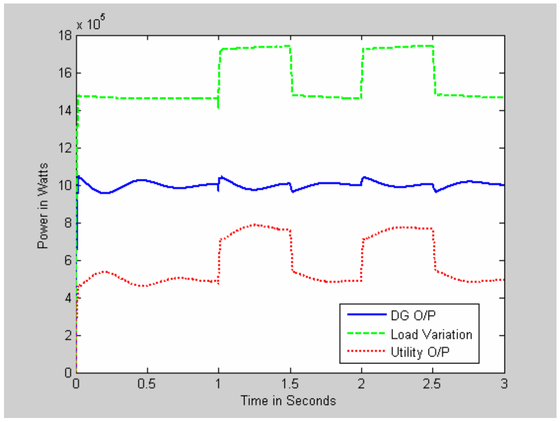
1. The fixed load is made variable, changing at times t = 1.0, t = 1.5, t = 2, t = 2.5, increasing from one third full load to one third plus one tenth full load and further decreasing to one third full load. The response of the DG with varying load is shown in Figure 5.32 (a). The utility output varies considerably with the load change. The generator controllers act at the time instant of changing loads but cannot compensate fully for the varying load.

Table 5.11. Voltages and Currents from RDAP without DG

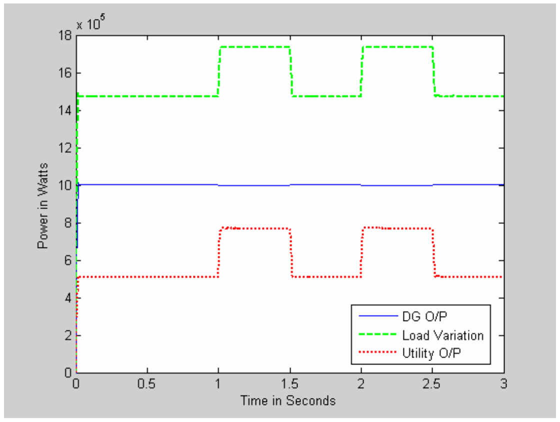
Bus No	|V| Volts	∠V (Deg)	Branch	|I| (Amp)	∠I (Deg)	Phase
1	7199.7	0	12	100.84	-32.45	A
1	7199.9	-120	12	130.92	-147.43	B
1	7199.6	120	12	164.52	97.52	C
2	7181	-0.05	23p	100.85	-32.46	A
2	7166	-120.12	23p	130.93	-147.44	B
2	7167.2	119.67	23p	164.52	97.52	C
3	2393.5	-0.06	23s	302.53	-32.46	A
3	2388.4	-120.13	23s	392.75	-147.44	B
3	2388.7	119.437	23s	493.53	97.52	C
4	2323.5	-0.67	34	302.53	-32.46	A
4	2262.3	-121.59	34	392.75	-147.44	B
4	2274.3	115.71	34	493.53	97.52	C

2. Now the DG is modeled as an independent current source. Figure 5.32 (b) shows the output of the DG and utility after independent current source interface. The utility supplies for the varying load. Now the PI controller is employed for providing information of changing load to the DG. Figure 5.33 shows that with autonomous control the utility output is constant, and load fluctuations are absorbed by the DG. The output of the DG is decided by $P_{DG}(t+1) = P_{DG}(t) + K_p \Delta P_L(t) + K_i \Delta P_L(t)$, where, ΔP_L is the change in load, P_{DG} is the real power output of the DG and $0.8MW \le P_{DG}(t) \le 2.5MW$ and

$P_{DG}(0) = 1.0$ MW. Placing a DG near the load reduces losses, improves reliability and substation power level is not affected.



(a)



(b)

Figure 5.32 (a) DG (Synchronous Generator) with Load Fluctuations (b) DG (Independent Current Source) with Load Fluctuations

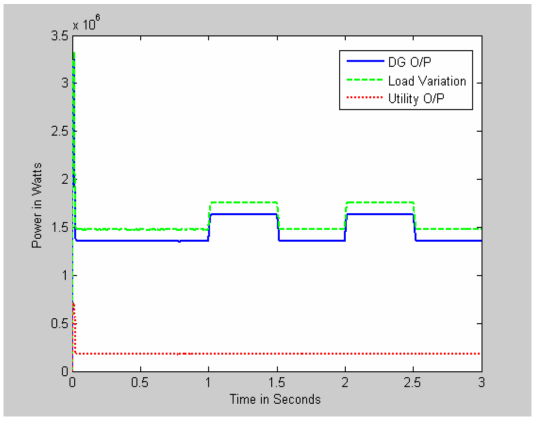


Figure 5.33 Control of DG (Independent Current Source) with Load Fluctuations

5.8 Computational Complexity

Figure 5.34 shows the complexity of the developed MAS for a distribution network by UML® class diagram. The class diagram is for restoration, load shedding and restoration through alternate path after islanding. This class diagram includes main package restoration and five kinds of agents whose classes are extended from the Agent class of JADE. The Simple Behavior and Delay Behavior classes of JADE were extended to realize the behaviors of different agents. JADE consists of a set of application program interface (API) classes and methods that allow an agent to perform the required actions of a Mobile Agent.

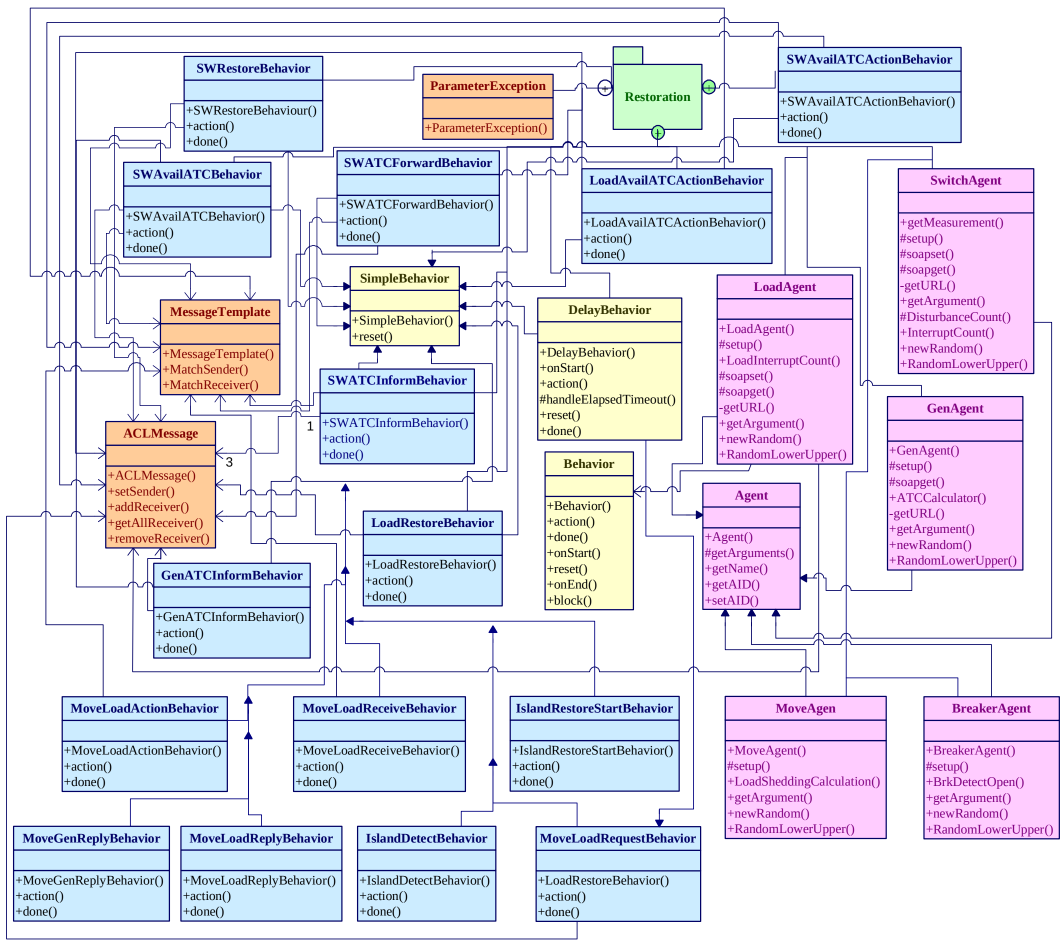


Figure 5.34 UML Class Diagram of Developed Multi-Agent System

5.9 Conclusions

This chapter enhances the restoration strategy presented in Chapter 4. Behaviors and agents were added to the MAS developed in Chapter 4 to perform restoration with DG. Islanding cases were considered, and the power balance is maintained in the island by load shedding performed using MAS. The load shedding considers priority, and non-critical/non-vital loads are shed first. After the load shedding the shed load agent sends a query for restoration. If an alternate path with sufficient capacity is available, the query for restoration is answered, and the load is restored. DG control plays an important role in restoration. A preliminary work on achieving autonomous control for DG to follow load changes is described. This control helps with fast settling of DG and increases the stability of the island with the added advantage of reducing losses.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

Deregulation, distributed generation and the ever increasing size of power systems have forced the need for fast, automated, intelligent and reliable load shedding and restoration schemes that can prevent system collapse. Careful study of systems with DG indicate a need for DG control in normal conditions and islanded conditions to augment the benefits of DG. The fast and powerful decentralized Multi-Agent System (MAS) technique developed in this research achieved the objective of identifying the amount and location of the load to be shed or restored.

6.1 Conclusions

This dissertation proposed the development and implementation of a fast, intelligent, automated and decentralized scheme for restoration and load shedding of land-based and ship distribution systems. The agents were first developed in MATLAB, and the system was simulated in VTB. VTB had limited three-phase models. The need of a three-phase unbalanced load led to the development of such a PQ load model. VTB has made function calls available for development of models required by the users. MATLAB-VTB co-simulation is performed for agent executions depending on the system measurements. The development is successfully tested on a test system.

Some drawbacks come with implementing a MAS in MATLAB, and hence the focus is on using JADE as the agent development platform. VTB-JADE communication was achieved, and an over-current fault was detected and isolated in real time. Three agents, SA, LA and GA, were developed, and they exchange information through their behaviors and determine a feasible restoration path. Three different test cases for full restoration, partial restoration with load shedding, and alternative path restoration demonstrated the implementation of the agents. The results indicate the MAS can provide a distributed solution for restoration strategies. The time required to make and implement decisions about load shedding or restoring is noted when the solution is implemented with the power system running. This time excludes the physical constraints.

This dissertation also develops a restoration scheme for systems with DG. For this focus, a DG model to the level of details required for restoration is developed in VTB. The model verification was done on a test system. Two agents were added and several behaviors were added to the agents existing in the MAS developed for restoration without DG. A Mobile Agent was added to obtain loading and generation from the LA and GA. The implementation of the agents was demonstrated using four distribution networks for load shedding within the island and load shedding within the island with restoration of shed loads through alternate paths having enough capacity. The results indicate the MAS can provide a distributed solution for maintaining islands and thereby increase the continuous supply to vital/critical loads. Since the fast variation of DG should not be assumed, research for autonomous control of DG was also achieved which helped in reducing losses and improving reliability; the substation power level was not affected. To implement this scheme, the DG was modeled as an independent current source. A PI controller was used to control the DG output to compensate for load variations.

The significant contributions of this work are:

- Development and implementation of a state-of-the-art Multi-Agent System
    - for restoration after isolation of fault, load shedding under islanded conditions
    - restoration of shed loads in the island using an alternate path
    - restoration with and without DG systems
- Autonomous control for DG to track load variations was designed.

The unique features of the developed MAS for restoration with and without DG are that it can be implemented with real time simulation, is decentralized completely, follows FIPA specifications, passes messages asynchronously and is modular. These developments are essential for fast reconfiguration for restoration of land-based and shipboard power systems with and without DG.

6.2 Future Work

The accomplishments of this research have posed several other problems that can be undertaken further. The MAS is a fairly new development for distribution system restoration and has not been previously considered with simulation of the system. So this scheme has much more potential to investigate. First, the autonomous control described in this dissertation can be implemented using agents to achieve speed and autonomy. Multi-tier restoration that considers transferring loads to an alternate feeder if the feeder available for restoration does not have enough capacity can be implemented with the addition of just several behaviors to existing agents. The power system can be simulated in PSCAD to study fast transients associated with switching actions. For this possibility, an interface between JADE and PSCAD needs to be developed. Further work would involve testing the restoration formulation on unbalanced systems. Also, centralized techniques based on mathematical programming and those based on heuristics and meta-heuristics should be compared with multi-agent solutions to demonstrate the effectiveness of the MAS scheme and the tradeoffs involved in other schemes. Since agents are software abstractions, the MAS is reusable, and hence making additions to incorporate restoration for an integrated (AC-DC) structure of ship system is recommended.

.

REFERENCES

[1] Ciezki J.G. and Ashton R.W., "Selection and Stability Issues Associated with a Navy Shipboard DC Zonal Electric Distribution System", *IEEE Transactions on Power Delivery*, vol. 15, no. 2, April 2000, pp. 665-669.

[2] Willis H. L., "Analytical Methods and Rules of Thumb for modeling DG-Distribution Interaction," *Proceedings of the IEEE Power Engineering Society Summer Meeting 2000*, vol. 3, July 2000, pp. 1643-1644.

[3] IEEE Standard for Interconnecting Distributed Resources with Electric Power Systems, IEEE Std. 1547, 2003.

[4] Daly Peter and Morrison Jay, "Understanding the Potential Benefits of Distributed Generation on Power Delivery Systems,"*Rural Electric Power Conference*, April-May 2001, pp. 2-1- 2-13.

[5] IEEE Standard Conformance Test Procedures for Equipment Interconnecting Distributed Resources with Electric Power Systems, IEEE Std. 1547.1, 2005.

[6] Basso S. Thomas and DeBlasio Richard, "IEEE 1547 Series of Standards: Interconnection Issues," *IEEE Transaction of Power Electronics*, vol. 19, no. 5, Sept. 2004, pp. 1159-1162.

[7] Foundations of Intelligent Physical Agents, http://www.fipa.org/

[8] Butler-Purry K. L., Sarma N. D. R. and I. V. Hicks, "Service Restoration in Naval Shipboard Power Systems," *IEE Proceedings on Generation, Transmission and Distribution*, vol. 151, no. 1, pp. 95-102, Jan. 2004.

[9] Khushalani S. and Schulz N. N., "Restoration Optimization With Distributed Generation Considering Islanding," *Proceeding of 2005 IEEE/PES General Meeting*, San Francisco, June 2005.

[10] Kyeong J., Park M., Kim H. S. and Jung-II S., "Development of Real-Time Service Restoration System for Distribution Automation System," *Proceedings of IEEE International Symposium on Industrial Electronics*, vol. 3, pp. 1514-1519, June 2001.

[11] Isamu W. and Makoto N., "A Genetic Algorithm for Optimizing Switching Sequence of Service Restoration in Distribution Systems," *Congress on Evolutionary Computation*, vol. 2, pp. 1683-1690, June 2004.

[12] Luan W. P., Irving M. R. and Daniel J. S., "Genetic Algorithm for Supply Restoration and Optimal Load Shedding in Power System Distribution Networks," *IEE Proceedings on Generation, Transmission and Distribution*, vol. 149, no. 2, pp. 145-151, March 2002.

[13] Fukuyama Y., "Reactive Tabu Search for Distribution Load Transfer Operation," *Proceedings of IEEE Power Engineering Society Winter Meeting*, vol. 2, pp. 1301-1306, Jan. 2000.

[14] Toune S., Fudo H., Genji T., Fukuyama Y. and Nakanishi Y., "A Reactive Tabu Search for Service Restoration in Electric Power Distribution Systems," *Proceedings on IEEE World Congress on Computational Intelligence – Evolutionary Computation*, pp. 763-768, May 1998.

[15] Mori H. and Ogita Y., "A Parallel Tabu Search Based Approach to Optimal Network Reconfiguration for Service Restoration in Distribution Systems," *Proceedings of the 2002 International Conference on Control Applications*, vol. 2, pp. 814-819, Sept. 2002.

[16] Jung K. H., Kim H. and Ko Y., "Network Reconfiguration Algorithm for Automated Distribution Systems Based on Artificial Intelligence," *IEEE Transaction on Power Delivery*, vol. 8, no. 4, pp. 1933-1941, Oct 1993.

[17] Liu C. C., Lee S. J. and Venkata S.S., "An Expert System Operational Aid for Restoration and Loss Reduction of Distribution Systems," *IEEE Transaction on Power Systems*, vol. 3, no. 2, pp. 619-626, May 1988.

[18] Tallat H. E. A., El-Safty S., Mansour M. M. and El-Debelky S., "A Rule-Based Expert System for Distribution System Service Restoration," *International Conference on PowerTech Budapest*, pp. 184, Sept. 1999.

[19] Hsiao Y. T. and Chien C. Y., "Enhancement of Restoration Service in Distribution Systems using a Combination Fuzzy-GA Method," *IEEE Transactions on Power Systems*, vol. 15, no. 4, pp. 1394-1400, Nov. 2000.

[20] Mendiola M. C., Chang C. S. and Elangoyan S., "Fuzzy Expert System for Distribution System Restoration and Contingency Operation," *International Conference on Energy Management and Power Delivery*, vol. 1, pp. 73-79, Nov. 1995.

[21] Sheng S., Sun Y., Liu Y., Zhang W. and Yang Y., "Integrating Genetic Algorithm with Expert System for Service Restoration in Distribution System," *International Conference on Power System Technology*, vol. 1, pp. 265-269, Aug. 1998.

[22] Mori H. and Furuta A., "A New Approach of Hierarchical Optimization to Distribution System Service Restoration," *IEEE International Symposium on Circuits and Systems*, pp. 4751-4754, May 2005.

[23] Mori H. and Tani H., "A Hybrid Method of PTS and Ordinal Optimization for Distribution System Service Restoration," *IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3476-3483, Oct 2003.

[24] Odell J., "Designing Agents: Using Life as a Metaphor," Distributed Computing, pp. 51–56, July 1998.

[25] Jennings N. R., Corera J. M. and Laresgoiti I., "Developing Industrial Multi-Agent Systems," *Proceedings of the First International Conference on Multi-agent Systems*, pp. 423-430, 1995.

[26] Wooldridge M. J. and Jennings N. R., "Pitfalls of Agent-Oriented Development," *Proceedings of 2nd International Conference on Autonomous Agents*, pp. 385-391, 1998.

[27] Yang J., Montakhab M., Pipe A. G., Carse B. and Davies T. S., "Application of Multi-Agent Technology to Fault Diagnosis of Power Distribution Systems," *Proceedings of the 4th International ICSC Symposium on Engineering of Intelligent Systems*, March 2004.

[28] Baran M., Sreenath R. and Mahajan N. R., "Extending EMTP for Simulating Agent Based Distributed Applications," *Proceedings of 14th Power System Computation Conference*, no. 32, pp. 1-4, June 2002.

[29] Takaya S., Katsuhiko S., Tatsuji T. and Katayama S., "A Remote Supervisory System for A Power System Protection and Control Unit Applying Mobile Agent Technology," *Proceedings of the IEEE/PES Transmission and Distribution Conference and Exhibition*, vol. 1, pp. 148-153, Oct. 2002.

[30] Wang X. R., Hopkinson K. M., Thorp J. S., Giovanini R., Birman K. and Coury D., "Developing an Agent-Based Backup Protection System for Transmission Networks," *Proceedings of the First International Conference on Power Systems and Communication Systems Infrastructures for the Future*, Beijing, China, Sept. 2002.

[31] Yu Y. M. and Li Z. Y., “Application of Multi-Agent System for Adaptive Protection System,” http://www.paper.edu.cn/scholar/download.jsp?file=zhuyongli-8

[32] Coury D. V., Thorp J. S., Hopkinson K. M. and Birman K. P., “Improving The Protection of EHV TEED Feeders Using Local Agents,” *Proceedings of the 7th International Conference on Developments in Power System Protection*, no. 479, pp. 527-530, April 2001.

[33] Wong S. K. and Kalam A., “An Agent Approach to Designing Protection Systems,” *Proceedings of the 6th International Conference on Developments in Power System Protection*, no. 434, pp. 373-376, March 1997.

[34] Tomita Y., Fukui C., Kudo H., Koda J. and Yabe K., “A Cooperative Protection System with an Agent Model,” *IEEE Transactions on Power Delivery*, vol. 13, no. 4, pp. 1060-1066, Oct. 1998.

[35] Esmin A. A. A., Aoki A., Lopes C. R. Jr. and Torres G. L., “Multi-Agent Simulation and Education Tool for Power System Operation,” http://www.lactec.org.br/publicacoes/2001/042_2001.pdf

[36] Buse D. P., Sun P., Wu Q. H. and Fitch J., “Agent-based Substation Automation,” *IEEE Power and Energy Magazine*, vol. 1, no. 2, pp. 50-55, April 2003.

[37] Sun L. H. and Cartes D., “Reconfiguration of Shipboard Radial Power System using Intelligent Agents,” *Proceedings of the Electric Machine Technology Symposium*, Philadelphia, Jan. 2004.

[38] Amin M. and Ballard D., “Defining New Markets for Intelligent Agents,” *IT Professional*, vol. 2, no. 4, pp. 29-35, Aug. 2000.

[39] Nagata T., Tao Y., Sasaki H. and Fujita H., “A Multi-Agent approach to Distribution System Restoration,” *Proceedings of 2003 IEEE/PES General Meeting*, Toronto, Canada, July 2003.

[40] Nagata T., Tao Y., Tahara Y., Aoyama T., Fujita H. and Koaizawa M., “Development of Bulk Power System Restoration Simulator by Means of Multi-Agent Approach,” *Proceedings of 47th IEEE International Midwest Symposium on Circuits and Systems*, vol. 2, pp. 337-340, IEEE 2004.

[41] Nagata T., Nakayama H., Utatani M. and Sasaki H.,“A Multi-Agent Approach to Power System Normal State Operation,” *Proceeding of 2002 IEEE/PES General Meeting*, vol. 3, pp. 1582-1586, Chicago, July 2002.

[42] Wang H., Schulz N. N., Cartes D., Sun L. H. and Srivastava S., "System Reconfiguration Strategy for Shipboard Power Systems Using Multi-Agent Systems," *Proceedings of the ASNE Reconfiguration and Survivability Symposium*, Florida, Feb. 2005.

[43I] Seca L., Pecas Lopes J.A., "Intentional islanding for reliability improvement in distribution networks with high DG penetration" *International Conference on Future Power Systems*, Nov. 2005, pp. 1-5.

[44] Choi Joon-Ho, Jae-Chul Kim, Seung-Il Moon, "Integration operation of dispersed generations to automated distribution networks for network reconfiguration," *IEEE Proceedings of Bologna Power Tech Conference*, vol. 2, pp. 23-26, June 2003.

[45] Flexible High Speed Load Shedding using a Crosspoint Switch,http://www.selinc.com/techpprs/6216_FlexibleLoad_WA_20050928.pdf

[46] Wong K.P. and Lau B.S., "Algorithm for load-shedding operations in reduced generation periods", *IEE Proceedings on Generation, Transmission*, Vol. 139, Issue 6, Nov 1992, pp. 478 – 490.

[47] Rao P.S.N., Rao K.S.P., "An efficient load shedding algorithm for radial systems," *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region*, vol. 2,Oct. 2003, pp. 771-774

[48] Chen, B.K., Chu W.C., Chen D.N.; Chen J.K., "Underfrequency load-shedding scheme for co-generators connected to an unreliable utility system," *IEEE Industry Applications Society Annual Meeting*, vol. 2, Oct. 1992, pp. 1261-1267.

[49] Shervin Shokooh et. al, "Intelligent Load Shedding Need for a Fast and Optimal Solution", http://www.etap.com/downloads/papers/ILS_Fast_Optimal.pdf

[50] Al-Hinai A. and Feliachi A., "Application of intelligent control agents in power systems with distributed generators," *Proceedings of the IEEE Power Systems Conference and Exposition*, vol.3, 10-13 Oct. 2004, pp. 1514 - 1519

[51] Zoka Y., Sasaki H., Yorino N., Kawahara K. and Liu, C.C.," An interaction problem of distributed generators installed in a MicroGrid," Proceedings of the IEEE Electric Utility Deregulation, Restructuring and Power Technologies, vol. 2, 5-8 April 2004, pp.795 – 799.

[52] Al-Hinai A., Sedhisigarchi K. and Feliachi A., "Stability enhancement of a distribution network comprising a fuel cell and a microturbine," IEEE Power Engineering Society General Meeting, vol. 2, 6-10 June 2004, pp. 2156 – 2161.

[53] Hara R., Muranaka T., Kita H., Tanaka, E. and Hasegawa, J., "Multi agent based emergency operation of quality control center considering uncooperative situation," IEEE/PES Transmission and Distribution Conference and Exhibition, Asia Pacific, vol. 2, 6-10 Oct. 2002, pp. 1185 – 1190.

[54] Hiyama T., Zuo D. and Funabashi T., "Multi-Agent Based Control and Operation of Distribution System with Dispersed Power Sources," *Proceedings of IEEE/PES Transmission and Distribution Conference and Exhibition*, vol. 3, pp. 2129-2133, Oct. 2002.

[55] Okuyama K. and Kato T., Suzuoki Y. and Funabashi T., "Improvement of Reliability of Power Distribution Systems by Information Exchange between Distributed Generators," *Proceedings of the 2001 IEEE/PES Summer Meeting*, vol. 1, pp. 468-473, July 2001.

[56] Teems E. L. and Eugene V. S., "Resistive Companion Modeling and Simulation for the Virtual Test Bed", http://vtb.engr.sc.edu/

[57] Shen W., D. Norrie and Barthes J. A., **Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing**, Taylor and Francis: London and New York, pp. 41-48, 2001.

[58] JADE Tutorial – JADE Programming for Beginners, http://jade.tilab.com/doc/JADEProgramming-Tutorial-for-beginners.pdf

[59] Sun L., Morejon G., and Cartes D., "Interfacing Software Agents with Power System Simulations," *Proceedings of the WSEAS Conference on Electroscience and Engineers and Technology for Naval Engineering and All-Electric Ship*, July 2004.

[60] Taylor W. Carson, **Power System Voltage Stability**, McGraw Hill, Inc., pp. 50, 1994.

[61] T.L. Baldwin and S.A. Lewis, **"**Distribution Load Flow Methods for Shipboard Power Systems," *IEEE Transactions on Industry Applications*, vol. 40, no. 5, pp. 1183 – 1190, Sept.-Oct. 2004.

[62] W.H Kersting, "Radial Distribution Test Feeders", *Proceedings of IEEE PES Winter Meeting*, Vol.2, pp. 908 –912, 2001

[63] RDAP User Manual, Version 3.0: Sept 1999, WH Power Consultants, Las Cruces-NM, http://www.zianet.com/whpower

APPENDIX

AGENT INITIALIZATIONS

Agents require local knowledge to make decisions. This local knowledge involving rpc ID, upstream and downstream neighbors, rated capacity, status and priority, is given in the agents' initialization. SWA, GA, LA, BA and MA initializations for the test cases discussed in Chapters 4 and 5 are given here.

1. Initialization for Test System I in Section 4.5.1.

java jade.Boot S1: Restore. SWA(1 1 100 1 S2 1 G1 0 1 L1) S2: Restore. SWA(2 1 100 1 S3 1 S1 1 L2 0) S3: Restore. SWA(3 1 100 1 S4 1 S2 0 1 L2) S4: Restore. SWA(4 1 100 1 S5 1 S3 1 L3 0) S5: Restore. SWA(5 1 100 1 S6 1 S4 0 1 L3) S6: Restore. SWA(6 1 100 1 S7 1 S5 0 0) S7:Restore. SWA(7 0 100 1 G2 1 S6 0 0) L1:Restore.LA(100 N 1 S1 1 G1) L2: Restore.LA(110 V 1 S3 1 S2) L3:Restore.LA(120 N 1 S5 1 S4) G1: Restore.GA(8 100 2 S1 L1) G2: Restore.GA(9 40 1 S7)

2. Initialization for Test System II in Section 4.5.2.

java jade.Boot S10:Restore.SWA(10 1 200 1 S11 1 G1 0 1 L1) S11:Restore.SWA(11 1 200 1 S12 1 S10 1 L2 0) S12:Restore.SWA(12 1 200 1 S13 2 S11 S16 0 1 L2) S13:Restore.SWA(13 1 200 1 S14 1 S12 1 L3 0) S14:Restore.SWA(14 1 200 1 S15 1 S13 0 1 L3) S15:Restore.SWA(15 1 200 1 S34 1 S14 0 0) S34:Restore.SWA(34 0 200 1 S27 1 S15 0 0) S16:Restore.SWA(16 1 200 1 S12 1 S17 1 L2 0 1) S17:Restore.SWA(17 1 200 1 S16 1 S18 0 1 L4) S18:Restore.SWA(18 1 200 1 S17 1 S19 1 L4 0) S19:Restore.SWA(19 1 200 1 S18 1 S20 0 1 L5) S20:Restore.SWA(20 1 200 1 S19 1 S21 1 L5 0) S21:Restore.SWA(21 1 200 1 S20 0 0 1 L6) L1:Restore.LA(100 N 1 S10 1 G1) L2:Restore.LA(110 V 1 S12 2 S11 S16) L3:Restore.LA(120 N 1 S14 1 S13) L4:Restore.LA(140 N 1 S17 1 S18) L5:Restore.LA(150 N 1 S19 1 S20) L6:Restore.LA(160 V 1 S21 0)

java jade.Boot -container S22:Restore.SWA(22 1 200 1 S23 1 G2 0 1 L7) S23:Restore.SWA(23 1 200 1 S24 1 S22 1 L8 0) S24:Restore.SWA(24 1 200 1 S25 2 S23 S28 0 1 L8) S25:Restore.SWA(25 1 200 1 S26 1 S24 1 L9 0) S26:Restore.SWA (26 1 200 1 S27 1 S25 0 1 L9) S27:Restore.SWA(27 1 200 1 S34 1 S26 0 0) S28:Restore.SWA(28 1 200 1 S24 1 S29 1 L8 0 1) S29:Restore.SWA(29 1 200 1 S28 1 S30 0 1 L10) S30:Restore.SWA(30 1 200 1 S29 1 S31 1 L10 0) S31:Restore.SWA(31 1 200 1 S30 1 S32 0 1 L11) S32:Restore.SWA(32 1 200 1 S31 1 S33 1 L11 0) S33:Restore.SWA(33 1 200 1 S32 0 0 1 L12) L7:Restore.LA(170 N 1 S22 1 G2) L8:Restore.LA(180 V 1 S24 2 S23 S28) L9:Restore.LA(190 N 1 S26 1 S25) L10:Restore.LA(200 N 1 S29 1 S30) L11:Restore.LA(210 N 1 S31 1 S32)

L12:Restore.LA(220 V 1 S33 0) G1:Restore.GA(4 550 2 S10 L1) G2:Restore.GA(5 550 2 S22 L7).

3. Iniitialization for Test System III in Section 4.5.3

java jade.Boot S1:Restore.SWA(1 1 100 1 S2 1 G1 0 1 L1) S2:Restore.SWA(2 1 100 1 S3 1 S1 1 L2 0) S3:Restore.SWA(3 1 100 1 S4 1 S2 0 1 L2) S4:Restore.SWA(4 1 100 3 S20 S26 S28 1 S3 0 0) S5:Restore.SWA(5 1 100 1 S6 1 G2 0 1 L3) S6:Restore.SWA(6 1 100 1 S7 1 S5 1 L4 0) S7:Restore.SWA(7 1 100 1 S8 1 S6 0 1 L4) S8:Restore.SWA(8 1 100 3 S20 S24 S27 1 S7 0 0) S9:Restore.SWA(9 1 100 1 S10 1 G3 0 1 L5) S10:Restore.SWA(10 1 100 1 S11 1 S9 1 L6 0) S14:Restore.SWA(14 1 100 1 S15 1 S10 0 1 L6) S15:Restore.SWA(15 1 100 3 S24 S25 S26 1 S14 0 0) S16:Restore.SWA(16 1 100 1 S17 1 G4 0 1 L7) S17:Restore.SWA(17 1 100 1 S18 1 S16 1 L8 0) S18:Restore.SWA(18 1 100 1 S19 1 S17 0 1 L8) S19:Restore.SWA(19 1 100 3 S25 S27 S28 1 S18 0 0) S20:Restore.SWA(20 0 150 1 S8 1 S4 0 0) S24:Restore.SWA(24 0 150 1 S15 1 S8 0 0) S25:Restore.SWA(25 0 150 1 S19 1 S15 0 0) S26:Restore.SWA(26 0 150 1 S15 1 S4 0 0) S27:Restore.SWA(27 0 150 1 S19 1 S8 0 0) S28:Restore.SWA(28 0 150 1 S19 1 S4 0 0) L1:Restore.LA(300 N 1 S1 1 G1) L2:Restore.LA(310 N 1 S3 1 S2) L3:Restore.LA(320 V 1 S5 1 G2) L4:Restore.LA(330 N 1 S7 1 S6) L5:Restore.LA(340 N 1 S9 1 G3) L6:Restore.LA(350 V 1 S14 1 S10) L7:Restore.LA(360 V 1 S16 1 G4) L8:Restore.LA(370 N 1 S18 1 S17) G1:Restore.GA(6 120 2 S1 L1) G2:Restore.GA(7 100 2 S5 L3) G3:Restore.GA(8 120 2 S9 L5) G4:Restore.GA(9 200 2 S16 L7)

4. Initialization for Test System I in Section 5.6.1.

java jade.Boot GA01:dgproject.GA(7 20 1 SA01) DA01:dgproject.GA(8 300 1 SA09) LA01:dgproject.LA(110 N 1 SA01 0) LA02:dgproject.LA(120 V 1 SA03 1 SA02) LA03:dgproject.LA(130 V 1 SA06 1 BA05) LA04:dgproject.LA(140 N 1 SA08 1 SA07) BA05:dgproject.BrkAgent(5 1 1 MA01) MA01:dgproject.MoveAgent(1 2 GA01 DA01 4 LA01 LA02 LA03 LA04) SA01:dgproject.SWA(1 1 200 1 SA02 1 GA01 0 1 LA01) SA02:dgproject.SWA(2 1 200 1 SA03 1 SA01 1 LA02 0) SA03:dgproject.SWA(3 1 200 1 SA04 1 SA02 0 1 LA02) SA04:dgproject.SWA(4 1 200 1 BA05 1 SA03 0 0) SA06:dgproject.SWA(6 1 200 1 SA07 1 BA05 0 1 LA03) SA07:dgproject.SWA(7 1 200 1 SA08 1 SA06 1 LA04 0) SA08:dgproject.SWA(8 1 200 1 SA09 1 SA07 0 1 LA04) SA09:dgproject.SWA(9 1 200 1 DA01 1 SA08 0 0)

5. Initialization for Test System II (Case 1) in Section 5.6.1.

java jade.Boot GA01:dgproject.GA(6 300 1 SA02) DA01:dgproject.GA(7 20.50 1 SA07) DA02:dgproject.GA(8 20.20 1 SA15) DA03:dgproject.GA(9 20.13 1 SA17) LA01:dgproject.LA(110 N 1 SA04 0) LA02:dgproject.LA(120 N 1 SA06 0)

LA03:dgproject.LA(130 V 1 SA09 0) LA04:dgproject.LA(140 N 1 SA12 0) LA05:dgproject.LA(150 N 1 SA14 0) LA06:dgproject.LA(160 V 1 SA19 0) BA01:dgproject.BrkAgent(1 1 1 MA01) BA11:dgproject.BrkAgent(11 1 1 MA01) MA01:dgproject.MoveAgent(1 4 GA01 DA01 DA02 DA03 6 LA01 LA02 LA03 LA04 LA05 LA06) SA02:dgproject.SWA(2 1 200 1 SA03 1 GA01 0 0) SA03:dgproject.SWA(3 1 200 2 SA08 BA11 1 SA02 0 0) SA04:dgproject.SWA(4 1 200 1 SA05 0 1 LA01 0) SA05:dgproject.SWA(5 1 200 2 SA8 BA11 1 SA04 0 0 1) SA06:dgproject.SWA(6 1 200 2 SA08 BA11 0 1 LA02 0 1) SA07:dgproject.SWA(7 1 200 1 SA08 1 DA01 0 0) SA08:dgproject.SWA(8 1 200 1 BA11 1 SA07 0 0) SA09:dgproject.SWA(9 1 200 1 SA10 0 1 LA03 0) SA10:dgproject.SWA(10 1 200 1 BA11 1 S09 0 0 1) SA12:dgproject.SWA(12 1 200 1 SA13 0 1 LA04 0) SA13:dgproject.SWA(13 1 200 2 SA16 SA18 1 SA12 0 0 1) SA14:dgproject.SWA(14 1 200 2 SA18 SA16 0 0 1 LA05 1) SA15:dgproject.SWA(15 1 200 1 DA02 1 SA16 0 0) SA16:dgproject.SWA(16 1 200 1 SA15 0 0 0) SA17:dgproject.SWA(17 1 200 1 DA03 1 SA18 0 0) SA18:dgproject.SWA(18 1 200 1 SA17 0 0 0) SA19:dgproject.SWA(19 1 200 1 SA20 0 0 1 LA06) SA20:dgproject.SWA(20 1 200 2 SA18 SA16 1 SA19 0 0 1)

6. Initialization for Test System I in Section 5.6.2.

java jade.Boot GA01:dgproject.GA(7 300 1 SA01) DA01:dgproject.GA(8 25 1 SA08) GA02:dgproject.GA(9 100 1 SA10) LA01:dgproject.LA(110 N 1 SA01 1 GA01) LA02:dgproject.LA(120 V 1 SA03 1 SA02) LA03:dgproject.LA(130 N 1 SA08 1 SA06) LA04:dgproject.LA(140 V 1 SA07 0) LA05:dgproject.LA(150 V 1 SA10 1 GA02) LA06:dgproject.LA(160 N 1 SA13 1 SA12) BA05:dgproject.BrkAgent(5 1 1 MA01) SA01:dgproject.SWA(1 1 200 1 SA02 1 GA01 0 1 LA01) SA02:dgproject.SWA(2 1 200 1 SA03 1 SA01 1 LA02 0) SA03:dgproject.SWA(3 1 200 1 SA04 1 SA02 0 1 LA02) SA04:dgproject.SWA(4 1 200 1 BA05 1 SA03 0 0) SA06:dgproject.SWA(6 1 200 1 SA08 1 SA07 1 LA03 0 1) SA07:dgproject.SWA(7 1 200 1 SA06 0 1 LA04) SA08:dgproject.SWA(8 1 200 1 SA09 2 SA06 BA05 0 1 LA03) SA09:dgproject.SWA(9 1 200 1 SA13 1 SA08 0 0) SA10:dgproject.SWA(10 1 200 1 SA12 1 GA02 0 1 LA05) SA12:dgproject.SWA(12 1 200 1 SA13 1 SA10 1 LA06 0) SA13:dgproject.SWA(13 0 200 1 SA09 1 SA12 0 1 LA06) MA01:dgproject.MoveAgent(1 2 GA01 DA01 4 LA01 LA02 LA03 LA04)

7. Initialization for Test System II in Section 5.6.2.

java jade.Boot GA01:dgproject.GA(6 300 1 SA02) DA01:dgproject.GA(7 22 1 SA16) DA02:dgproject.GA(8 22 1 SA14) GA02:dgproject.GA(9 300 1 SA22) LA01:dgproject.LA(110 V 1 SA04 0) LA02:dgproject.LA(120 N 1 SA07 0) LA03:dgproject.LA(130 N 1 SA09 0) LA04:dgproject.LA(140 V 1 SA32 0) LA05:dgproject.LA(150 V 1 SA24 0) LA06:dgproject.LA(160 V 1 SA27 1 SA26) BA01:dgproject.BrkAgent(1 1 1 MA01) BA06:dgproject.BrkAgent(6 1 1 MA01) SA02:dgproject.SWA(2 1 200 1 SA03 1 GA01 0 0) SA03:dgproject.SWA(3 1 200 2

SA29 BA06 1 SA02 0 0) SA04:dgproject.SWA(4 1 200 1 SA05 0 0 1 LA01) SA05:dgproject.SWA(5 1 200 2 SA29 BA06 1 SA04 0 0 1) SA07:dgproject.SWA(7 1 200 1 SA08 0 0 1 LA02) SA08:dgproject.SWA(8 1 200 3 SA15 SA17 SA18 1 SA07 0 0 1) SA09:dgproject.SWA(9 1 200 3 SA15 SA17 SA18 0 0 1 LA03 1) SA14:dgproject.SWA(14 1 200 1 SA15 1 DA02 0 0) SA15:dgproject.SWA(15 1 200 1 SA18 1 SA14 0 0) SA16:dgproject.SWA(16 1 200 1 SA17 1 DA01 0 0) SA17:dgproject.SWA(17 1 200 2 SA18 1 SA16 0 0) SA18:dgproject.SWA(18 1 200 1 SA19 3 SA17 SA15 BA06 0 0) SA19:dgproject.SWA(19 1 200 1 SA30 1 SA18 0 0) SA22:dgproject.SWA(22 1 200 1 SA23 1 GA01 0 0) SA23:dgproject.SWA(23 1 200 2 SA29 SA25 1 SA22 0 0) SA24:dgproject.SWA(24 1 200 2 SA29 SA25 1 SA23 0 0 1 LA05 1) SA25:dgproject.SWA(25 1 200 1 SA26 2 SA23 SA29 0 0) SA26:dgproject.SWA(26 1 200 1 SA27 1 SA25 1 LA06 0) SA27:dgproject.SWA(27 1 200 1 SA28 1 SA26 0 1 LA06) SA28:dgproject.SWA(28 1 200 1 SA30 1 SA27 0 0) SA29:dgproject.SWA(29 0 200 2 SA25 BA06 2 SA03 SA23 0 0) SA30:dgproject.SWA(30 0 200 1 SA19 1 SA28 0 0) SA32:dgproject.SWA(32 1 200 1 SA33 0 0 1 LA04) SA33:dgproject.SWA(33 1 200 3 SA15 SA17 SA18 1 SA32 0 0 1) MA01:dgproject.MoveAgent(1 3 GA01 GA02 GA03 4 LA01 LA02 LA03 LA04)