A MULTIAGENT SYSTEM APPROACH FOR A SELF-RECONFIGURABLE ELECTRIC POWER DISTRIBUTION SYSTEM

By

Janeth Gissella Gómez Gualdrón

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
COMPUTER ENGINEERING

University of Puerto Rico
Mayagüez Campus
2006

Approved by:

Agustín A. Irizarry Rivera, Ph.D. Member, Graduate Committee	Date
Manuel Rodríguez Martínez, Ph.D. Member, Graduate Committee	Date
Bienvenido Vélez Rivera, Ph.D. Member, Graduate Committee	Date
Miguel Vélez Reyes, Ph.D. President, Graduate Committee	Date
Didier M. Valdés Díaz, Ph.D. Representative of Graduate Studies	Date
Isidoro Couvertier, Ph.D. Chairperson of the Department	Date

ABSTRACT

Electric power distribution systems (EPDS) can be found almost everywhere, from ship power systems to data centers. In many critical applications, it is necessary to maintain minimal operating capabilities under fault conditions. Therefore, it is necessary to develop energy distribution control techniques, which allow the implementation of a self-reconfigurable EPDS. This research project focuses on the application of MultiAgent Systems (MAS) to develop a self-reconfigurable EPDS. MAS are composed of multiple interacting software elements, known as agents. An agent is an abstraction that describes autonomous software that "acts on behalf of" a user or another program. A prototype of a MAS is proposed to reconfigure an electric system in order to maximize the number of served loads with highest priority. We used three simulation test systems based on zonal architecture. The test systems were simulated using Matlab™ Simulink™ - SimPowerSystems toolbox, and the MAS was implemented using Java™ programming language and JADE platform.

RESUMEN

Los sistemas de distribución de potencia eléctrica se encuentran casi en todas partes, desde sistemas de potencia en barcos hasta centros de datos. En muchas aplicaciones críticas, es necesario mantener capacidades de funcionamiento mínimas bajo condiciones de falla. Para alcanzar este objetivo, se hace necesario desarrollar técnicas de control para distribución de potencia, las cuales permitan la implementación de un sistema auto-reconfigurable. Este proyecto de investigación está enfocado en la aplicación de los sistemas multiagente (MAS) para desarrollar un sistema auto-reconfigurable de distribución de potencia. Los MAS están compuestos por múltiples elementos de software, conocidos como agentes, los cuales interactúan entre ellos. Un agente es una abstracción que describe un software autónomo que "actúa en nombre" de un usuario o de otro programa. Se propone un prototipo software de un sistema multiagente que reconfigura un sistema de potencia buscando maximizar el número de cargas servidas tengan la mayor prioridad. Se usaron tres modelos de sistemas de potencia de barcos basados en arquitecturas zonales. Estos sistemas utilizados como casos de prueba, estan simulados en Matlab™ Simulink™ - SimPowerSystems toolbox y el sistema multiagente esta implementado usando el lenguaje de programación Java y la plataforma de agentes JADE.

To God,

To Luis, for walking with me,

To the people strong enough to take his/her things, go away and sail through the seas of life in search of destiny... And the ones who taught me how to do it!

ACKNOWLEDGMENTS

First, I would like to thank my Advisor, Professor Miguel Vélez Reyes for all his support, patience, and guidance all the time. Thanks for letting me work under your supervision. I will always admire you and respect you as a professor, as a professional and as an excellent person.

I would like to thank Dr. Bienvenido Vélez, Dr. Manuel Rodríguez, Dr. Agustín Irizarry and Dr. Didier Valdés for serving as members of my graduate committee.

Thanks to the Electrical and Computer Engineering Department faculty and staff. Thanks to the Electric Power Networks Efficiency and Security (EPNES) project and to the Center for Power Electronics Systems (CPES). I want to thank all my research partners, "EPNES Kids": Christian, Idálides, Marianela, Carlos, and Alfredo. Thanks also to all my master partners and my friends from CPES, CENSSIS, ADMG and PDC. Thanks to all of the people who directly or indirectly have helped through my graduate period.

Thanks to my family, for teaching me that everything is earned by working hard and with sacrifice. Thanks to my best friends for offering their friendship and motivation every moment, even when we do not talk so much. Especially thanks to Iván, Luis and Gustavo for reading my thesis and for being my "English gurus", I know it was a big sacrifice.

I would like to thank my love Luis, for always giving me support, for always giving me love. For supporting me and understanding me when I was working late in the lab, especially when the system was not working and I was angry all the time. Thanks for walking next to me.

Lastly, but most importantly, I want to thank God for giving me the health, strength, and motivation to finish this dream. Thanks God for all the above, for giving me the opportunity to feel every day how much you love me.

This work was primarily supported by the National Science Foundation under grant ECS-0224743. This work made use of ERC shared facilities supported by the National Science Foundation under Award Number EEC-9731677.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Justification

Electric power distribution systems (EPDS) can be found almost everywhere, in data centers, automobiles, ships and aircrafts, all of which require sophisticated control techniques to support all aspects of operation including failures. In many critical applications such as navy ships and server rooms, they need to maintain minimal operating conditions under failure conditions. When a high degree of survivability is desired the effects of failures must be mitigated and control must be maintained in survivable scenarios. To achieve this goal it is necessary to make a series of decisions and control actions:

- The fault has to be detected.
- The fault source has to be identified and its magnitude estimated (partial degradation vs. total failure).
- Depending on the nature of the failure, actions have to be taken to compensate it:
  - A new functional network topology is chosen.
  - The EPDS has to be reconfigured.

These decisions must be made by a self-reconfigurable control system that incorporates not only simple regulatory loops and the supervisory control logic, but also a set of components that detect, isolate, and manage faults, in coordination with the control functions. The goal is to increase survivability, eliminate human mistakes, make intelligent reconfiguration decisions more quickly, reduce the manpower required to perform the functions, and provide electric power to critical loads through the surviving system [1]. In fact, it is envisioned that in the future, the EPDS will have the intelligence for rapid self-reconfiguration under fault scenarios. The concept of a self-reconfigurable Electric Power Distribution System is shown in Figure 1.1, which presents the tasks and the control process in a self-reconfigurable system. This control could be centralized or distributed.

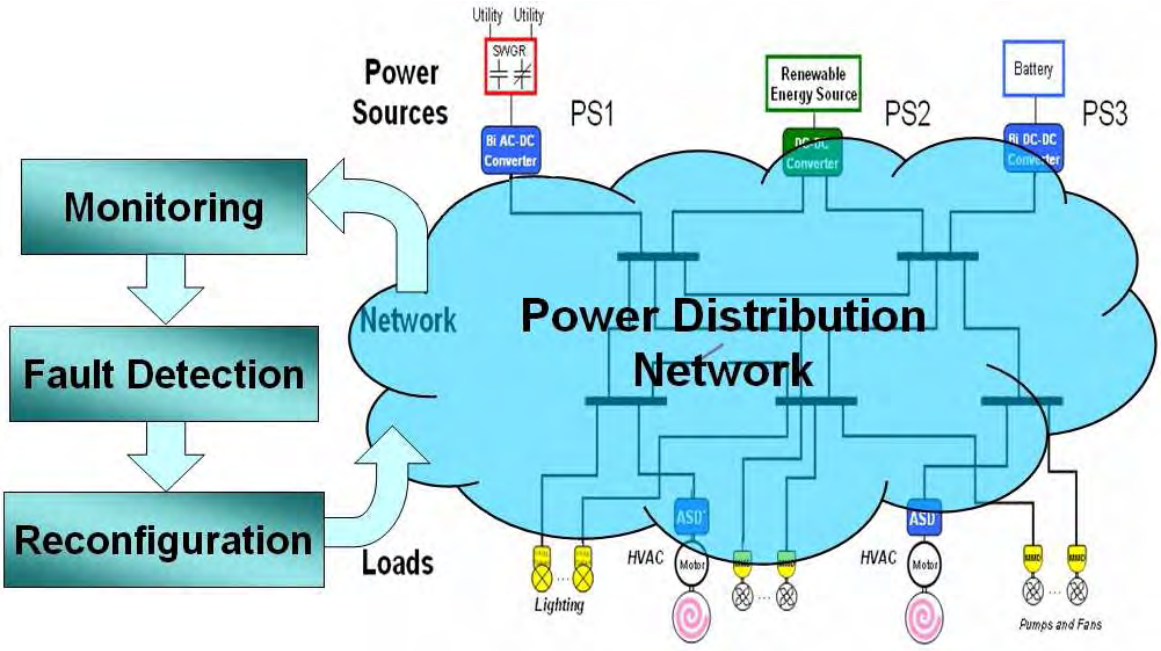


Figure 1.1: A Self-reconfigurable EPDS.

Traditionally, the control and coordination of power systems occurs in a centralized way and requires a considerable amount of human intervention [2]. In this scheme, only a few sites (or even one site) manage the mission-critical tasks in the electrical power system.

Various approaches have been proposed to control a power system in a centralized way, including[1] optimization and probabilistic methods, Genetic Algorithms, Artificial Neural Networks, Expert Systems, among others. But what happens if the failure affects the centralized controller? A critical situation is shown in Figure 2.1: in a fault event, a delayed answer or the absence of an action might result in a total collapse of the system [2, 3].

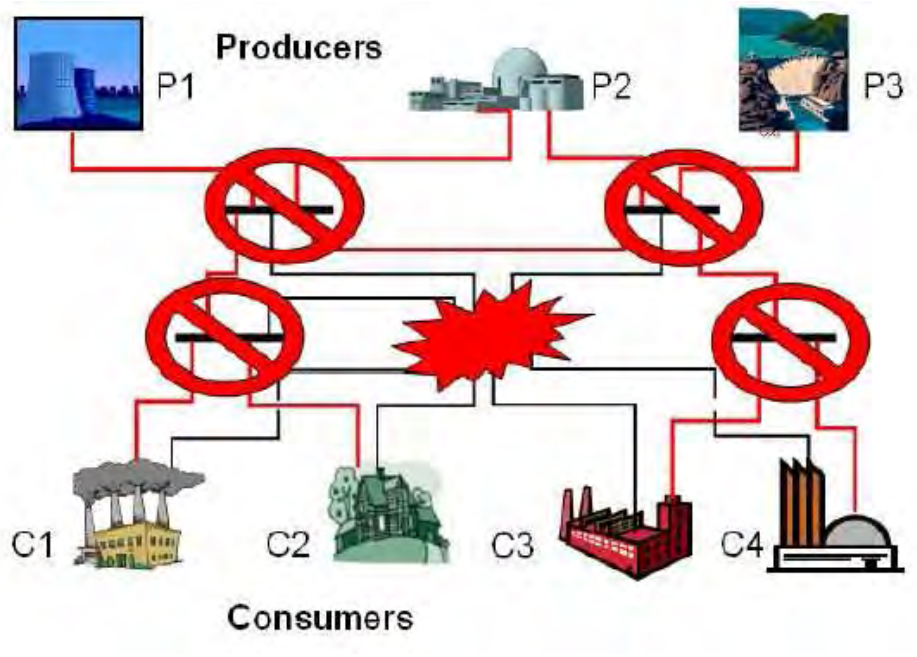


Figure 1.2: A slow response could cause a cascade failures [3].

For these reasons, the traditional centralized control is being replaced by Intelligent Distributed Controllers across the Electric Power Distribution System (EPDS), where the tasks of control and coordination are distributed across the system. In order to respond to the challenge of the new power systems, state-of-the-art software techniques must be used.

Some research has been dedicated to find the most suitable technology to develop the distributed control of an EPDS. In [4], Jennings and Bussman concluded that MultiA-

[1] More information about this methods will be presented in Chapter 2

gent Systems (MAS) are a well suited technology to develop complex software systems in general, and control systems in particular. MAS offers a high degree of scalability, independence of network topology, flexibility and robustness because they operate autonomously and make local decisions. MAS are a subfield of Distributed Artificial Intelligence and are composed of multiple interacting computing elements, known as agents. An agent is an abstraction, a logical model that describes software that "acts on behalf of" a user or other program. The software agent has the authority to decide if and when an action is appropriated, and it also has the autonomy to activate itself or be invoked by a task [5].

This research project focuses on the application of MultiAgent Systems (MAS) to develop a self-reconfigurable EPDS. The main idea is to design a self-reconfigurable EPDS using MAS in order to maximize the number of served loads with highest priority, for maintaining minimal operating conditions after a fault event.

1.2 Research Objectives

Our research is based on the concept of Intelligent Power Router (IPR), which was proposed by [2], as a part of the Electric Power Networks Efficiency and Security (EPNES) program sponsored by National Science Foundation (NSF). They proposed to develop a model for the next generation power network using a distributed concept based on a scalable coordination using the IPR. The general objective is to show that distributing the intelligence and the control functions over the network using the IPR, it is possible to improve survivability, security, reliability, and re-configurability [2].

Figure 1.3 shows the organization of the EPNES project at UPRM. According to this organization and the central goal of this research, this thesis is involved in the sections of the IPR Protocols and Distributed Control Models.

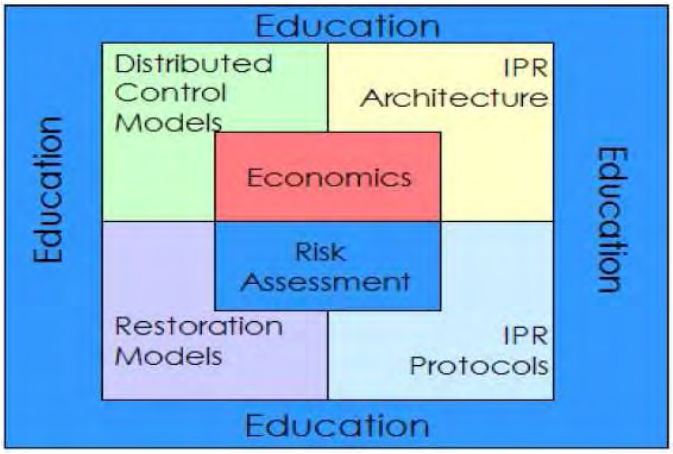


Figure 1.3: IPR Project structure [2].

The central goal of this research is to design a self-reconfigurable Electric Power Distribution System using MultiAgent Systems in order to maintain minimal operating conditions after a fault event. The agents in the system must be distributed across the EPDS. Each agent can negotiate and make decisions by communicating with other agents in the MAS.

In order to achieve this goal, some specific objectives must be accomplished:

1. Define the system architecture.
2. Design the MultiAgent System defining messages types, interaction protocols and behaviors of each type of agent.
3. Develop a functional prototype of the MultiAgent System based on the specifications obtained in the design phase.
4. Integrate the MultiAgent System with the simulation test system.
5. Evaluate the capabilities of the MultiAgent System-based reconfigurable electronic power distribution system.

1.3 Summary of Contributions

The major contributions of this thesis can be summarized as follows:

1. Designed the logic of a self-reconfigurable DC Zonal System using MultiAgent Systems. The principal characteristics of the proposed software framework are:
   - The architecture is designed to complement but not substitute existing control mechanisms and it includes the messages types, interaction protocols, behaviors for each agent and a negotiation scheme based on the Contract-Net Interaction Protocol.
   - The MultiAgent System is based on the FIPA specifications[2]to guarantee the interoperability of MAS.
   - The MAS is distributed, parallel and asynchronous, based on the Agent Model. Therefore, the MAS can be implemented using technologies, such as JAVA programming language and JADE platform, that allows the agent platform to be split on several host to implement a real distributed system.
2. Implemented a MAS prototype and integrated it with three simulation test systems. The test systems were simulated using SimPowerSystems^{TM} toolbox, which is part of Matlab^{TM} Simulink^{TM}. The MultiAgent System is developed using Java programming language and JADE platform.[3]
3. Implemented a procedure for interfacing JADE development platform and Matlab^{TM} Simulink^{TM}. We developed this interface, because the implementation of a real EPDS in order to evaluate a MultiAgent system is not cost-effective, and an alternative is to use a simulation package to model the EPDS. The interface allows the communication

[2] The Foundation for Intelligent Physical Agents (FIPA) is an IEEE Computer Society standards organization, which standardizes all the aspects related with agent technology and MAS [6]. The most relevant aspects of FIPA specifications for this work will be presented in Section 2.3.2 of this document.

[3] Java Agent DEvelopment Framework (JADE) is a JAVA framework for developing FIPA-compliant agent applications [7]. More information about JADE will be presented in Section 4.3.1 of this document.

between the simulation package and the MAS, which makes it a simple but useful tool for development and verification of control algorithms implemented with agents.

4. Demonstrated the usefulness of the MAS to reconfigure the EPDS in different scenarios. The obtained results show successful reconfiguration in the scenarios and test systems used.

5. Two papers were presented: one at the 10^{th} IEEE Workshop on Computers in Power Electronics (COMPEL06) [8] and another at the SPIE Defense and Security Symposium - Intelligent Computing: Theory and Applications Conference [9].

1.4 Thesis Structure

Chapter 2 provides a literary review and it is arranged in five subtopics: The Intelligent Power Routers (IPRs), MultiAgent Systems, Electronic Power Distribution Systems, Reconfiguration and Survivability of EPDS, and MAS applied to EPDS. Chapter 3 presents the information related to the design of the MAS describing the general architecture of the system, the scenarios in which the system operates, the types of agents, and the negotiation process. Chapter 4 provides insight on the implementation of the MAS prototype and its integration with the DC Zonal System. Chapter 5 presents the experiments and the obtained results. Finally, Chapter 6 presents the conclusions of the study and some recommendations for future work.

CHAPTER 2

Literature Review

2.1 Overview

This Chapter presents the literature review on the basic concepts of Intelligent Power Routers (IPRs), MultiAgent Systems, Electric Power Distribution Systems, and Reconfiguration and Survivability of EPDS. The Chapter concludes with a description of the main research that has been conducted on the application of MAS to System Reconfiguration.

2.2 The Intelligent Power Routers (IPRs)

The concept of Intelligent Power Router (IPR) was proposed by researchers at the University of Puerto Rico, Mayagüez. They proposed to develop a model for the next generation power network using a distributed concept based on scalable coordination by an IPR. The goal was to show that if the intelligence and control functions were distributed over the network using the IPR, it was possible to improve survivability, security, reliability, and re-configurability [2].

If a centralized control is used, a major disturbance to the system could cause a cascaded failure due to slow operator response, as shown in Figure 2.1 [2, 3]. This Figure was presented in Chapter 1 and it is repeated hereto present the approach.

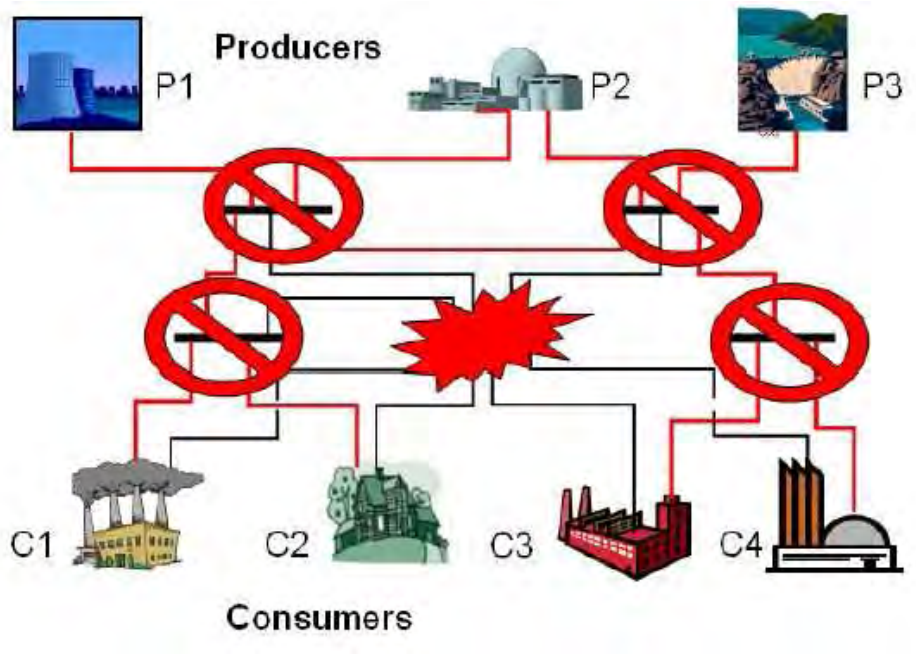


Figure 2.1: A slow response could cause cascade failures [3].

In the proposed system, the control is delegated to IPRs. They are strategically distributed over the entire electric network, instead of remaining in the main control center. Each IPR has embedded intelligence to switch power lines, shed load based on priorities, activate auxiliary or distributed generation, isolate the power region of the electric network to prevent cascaded failures, and exchange information with other IPRs. This last capability provides coordination between the IPRs for reconfiguring the network to avoid a cascade failure when facing major any natural or man-made disturbance [2]. Figure 2.2 presents the general vision of an IPRs-based system.

The idea of the IPR was motivated by data routers in data networks, where data is moved over geographically distant nodes through routers. An electric network distributes

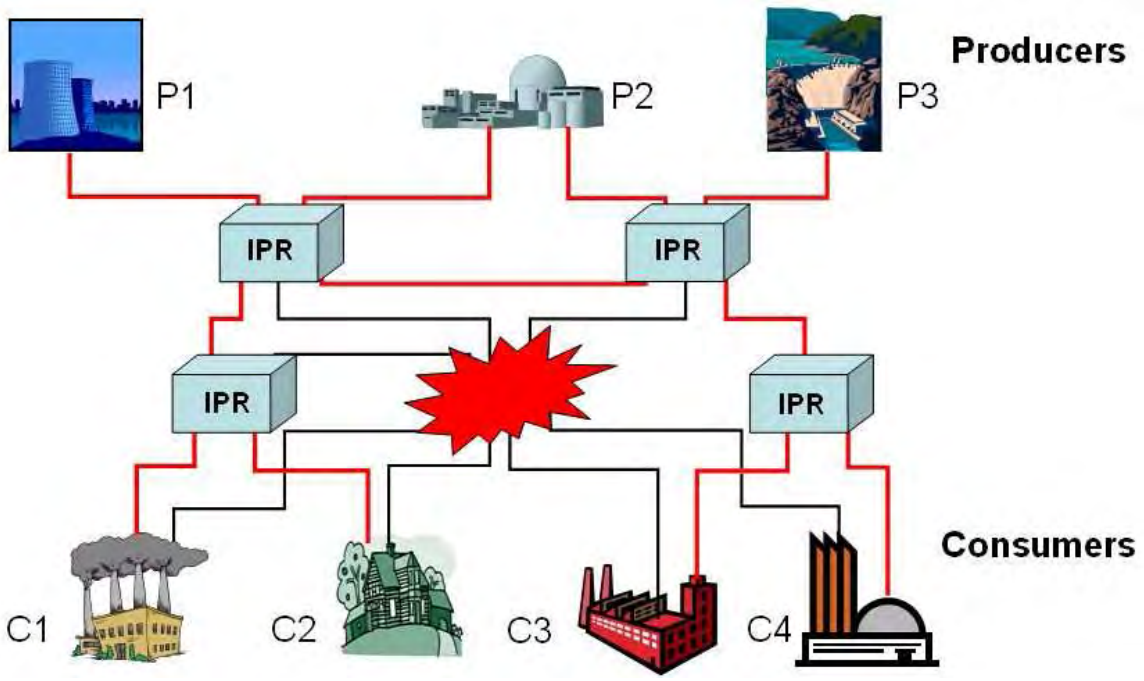


Figure 2.2: IPRs-based system [2].

energy over a region and the proposed IPR [2] is used to control the energy exchange in a distributed way. If a component or a system fails, the IPR will make local decisions, like a data router, and if needed, it will coordinate with neighbor IPRs to re-route energy flow necessary to maintain the system, or part of it operation [2].

2.2.1 IPR Proposed Architecture

The proposed architecture for an IPR is shown in Figure 2.3 [2]. It consists of interfacing circuits and an Intelligent Communication and Control Unit (ICCU). The ICCU receives information from sensors and it has the intelligence to reconfigure the system by using flow control devices under its control.

The work presented in this thesis deals with the intelligence of the IPR and here we presented an approach based on MAS.

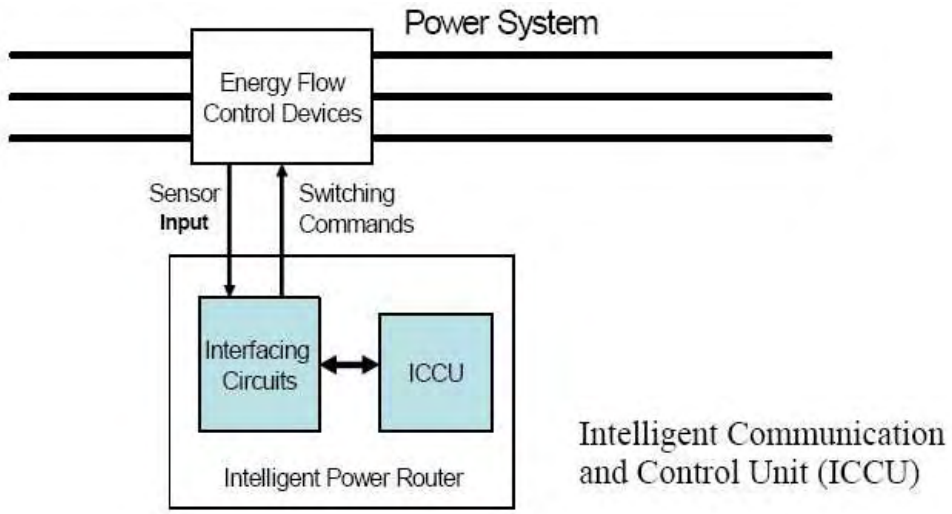


Figure 2.3: IPR proposed architecture [2].

2.3 MultiAgent Systems

MultiAgent Systems (MAS) are one of the most interesting new sub-fields of Computer Science and Distributed Artificial Intelligence. They have been studied since 1980, but the international interest in them grew rapidly after the mid-1990s. MAS are composed of multiple interacting computing elements, known as agents. As a general description, located in some environment, that reacts to the changes in this environment and it is capable of acting in order to achieve its goals [10].

2.3.1 Basic Concepts of Agents.

The concept of "agent" in computer science arose in the field of artificial intelligence in the mid-nineties. The essence of the agent concept is related to the property that marks its name, the agency. The "agency" is the property used to represent another organization: policeman, selling agent, purchasing agent. In order to wield this representation, an agent must have a minimum of autonomy. Thus the "agency" and the autonomy are the two

more important properties of an agent.

The agent concept combines aspects of diverse fields like economy, psychology, business administration, and artificial intelligence. Within this area, the concept of an agent evolved from concepts such as knowledge source, actors, daemon, concepts that date from the seventies [10, 11].

Given the multiplicity of roles agents can play, it has been impossible to provide a formal definition. Some authors have presented diverse definitions of the agent concept, each one emphasizing their own interest. Thus the definition of an agent varies based on their characteristics, requirements and technology.

Although there is no universally accepted definition of an agent; a list of general characteristics is provided [10]:

- Autonomy. An agent is autonomous when it operates without the direct intervention of humans or other agents. It can decide by itself what it needs to do in order to achieve its objectives.
- Social ability. An agent must have the capacity to cooperate, coordinate and negotiate with other agents using a communication language. This characteristic is an important aspect of agent societies.
- Reactivity. An agent perceives its environment and responds in a timely fashion to changes that occur in it.
- Proactiveness. The capacity to exhibit a goal-directed behavior by taking the initiative (it does not wait to receive an order) for advancing towards an objective.

The agents contained in MultiAgent environments are autonomous and distributed, and can be selfish or cooperative [11]. The software environment when agents run is called "Agent Platform". The aspects related with agent technology and MAS are standardized

by The Foundation for Intelligent Physical Agents (FIPA) [6].

2.3.2 The Foundation for Intelligent Physical Agents (FIPA)

The Foundation for Intelligent Physical Agents (FIPA) is an IEEE Computer Society standards organization. It was officially accepted by the IEEE as its eleventh standards committee in June 2005 [6]. FIPA was founded in 1996 as a Swiss not-for-profit organization to define a set of specifications for heterogeneous and interacting agents and agent-based systems. FIPA specifications represent a collection of standards which are intended to promote the inter-operation of heterogeneous agents and the services they can represent.

FIPA compliant systems must have certain characteristics:

- Agent Management provides the normative framework within which FIPA agents exist and operate.
- Agents communicate by exchanging messages which represent speech acts, and which are encoded in an Agent Communication Language

2.3.2.1 Agent Management Reference Model

Agent management provides the normative framework within which FIPA agents exist and operate. It consists of the following logical components:

- An Agent is the fundamental actor on an agent system and it must have at least one owner. An Agent Identifier (AID) labels an agent so that it may be distinguished unambiguously within the Agent Universe. An agent may be registered at a number of transport addresses at which it can be contacted and it may have certain resource brokering capabilities for accessing software.
- A Directory Facilitator (DF) is a mandatory component of the Agent Platform (AP). The DF provides yellow pages services to other agents. Agents may register their

services with the DF or query the DF to find out what services are offered by other agents.

- An Agent Management System (AMS) is a mandatory component of the AP. The AMS exerts supervisory control over access to and use of the AP. The AMS maintains a directory of AIDs which contain transport addresses for agents registered with the AP. The AMS offers white page services to other agents. Each agent must register with an AMS in order to get a valid AID.
- A Message Transport Service (MTS) is the default communication method between agents on different APs.
- An Agent Platform (AP) provides the physical infrastructure in which agents can be deployed. The AP consists of the machine(s), operating system, agent support software, FIPA agent management components (DF, AMS and MTS) and agents. The concept of an AP allows agents to reside on different host computers.
- Software describes all non-agent, executable collections of instructions accessible through an agent. Agents may access software, for example, to add new services, acquire new communications protocols, acquire new security protocols/algorithms, acquire new negotiation protocols, access tools which support migration, etc.

2.3.2.2 FIPA ACL

Communication is the most important feature for interaction between agents in MAS because it enables them to interact and share information for performing tasks to achieve their goals. Two most popular agent languages are Knowledge Query Manipulation Language (KQML) and FIPA Agent Communication Language (FIPA ACL), but the support for KQML has been discontinued in favor of FIPA ACL [12].

The FIPA Agent Communication Language (ACL) is based on speech act theory:

messages are actions, or communicative acts, as they are intended to perform some action by virtue of being sent. The specification consists of a set of message types and the description of their pragmatics [13].

The full set of FIPA ACL message parameters is shown in Table 2.1

Table 2.1: FIPA ACL Message Parameters

Parameter	**Category of Parameters**
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

Table 2.2 summarizes the full list of communicative acts (performatives) with the use of each of them.

The full set of parameters is available but designers can only use some of them, such as the following FIPA ACL message example:

```
(inform
    :sender Agent-A
    :receiver Agent-B
    :reply-with round01
    :content (price (bid good02) 100)
    :language fipa-sl
    :ontology auction
)
```

In this example, Agent-A informs Agent-B of the price offered during the first round of an auction. The first element of the message identifies the communicative act, the

Table 2.2: FIPA ACL Communicative Acts

Communicative Act	**Information passing**	**Requesting Information**	**Negotiation**	**Action performing**	**Error handling**
accept–proposal			√		
agree				√	
cancel				√	
cfp			√		
confirm	√				
disconfirm	√				
failure					√
inform	√				
inform-if	√				
inform-ref	√				
not-understood					√
propagate				√	
propose			√		
query-if		√			
query-ref		√			
refuse				√	
reject-proposal			√		
request				√	
request-when				√	
request-whenever				√	
subscribe		√			

principal meaning of the message (in this case "inform"). Then, the sequence of message parameters follows, introduced by parameter keywords beginning with a colon character.

2.3.2.3 FIPA Interaction Protocols

A conversation between agents has two roles, initiator and responder. For instance, an initiator agent makes a request to a responder agent (the initiator can make a request to more than one responder). Each responder can act on the request and then inform the initiator that the request has been performed, or it can refuse the request.

As you can see from the previous example, a conversation is composed of messages that can be seen as a sequence of performative acts such as REQUEST, INFORM or REFUSE. FIPA has standardized some conversations, known as the FIPA Interaction Protocols (IPs) [14]. Thus, a FIPA IP is a set of pre-agreed ACL messages and sequences used during the conversation.

FIPA has standardized eleven IP for several types of conversations between agents.

The complete specification of each conversation, including a sequence diagram, can be found in [14]. One of the most widely used protocols is the Contract Net Interaction Protocol, which was used for this research and will be explained when describing the MAS in the next chapter.

2.3.3 MultiAgent Systems Applications

Agent applications can be classified by the type of agent, by the technology used to implement the agent, or by the application domain itself. Next, we use the domain type as classification scheme [4][15].

2.3.3.1 Industrial Applications

Some examples of Industrial applications of agent technology are described next [15, 10]:

- Process Control. Agent technology is used in process control, because process controllers can be autonomous reactive systems. They have been applied in electric transportation management and particle accelerator control.
- Manufacturing. Some research applies the Contract Net protocol to manufacturing control. In order to manage the production process at plants efficiently. This process is defined by parameters that change, such as the products to be manufactured, available resources and time constraints. The Contract-Net protocol allows tasks to be delegated to individual factories, and from individual factories down to flexible manufacturing systems, and then to individual workcells.
- Air Traffic Control. Agents are used to represent aircrafts and the various air traffic control systems in operation. As an aircraft enters to the airport airspace, an agent

[4] More information about the applications of MultiAgent systems can be found in [15].

is allocated and instantiated with the information and goals corresponding to the real-world aircraft.

2.3.3.2 Commercial Applications.

The Commercial Applications of MAS have focused on three principal areas: Information Management, Electronic Commerce, and Business Process Management [15, 10]:

- Information Management. The lack of effective information management tools has risen due to the richness and diversity of information available. It is possible to characterize the information overload problem in two ways, information filtering and information gathering. The first one presents that there is an enormous amount of information available only part of which is relevant or important for a particular person. The later referes to the volume of information available, which makes it difficult to find information for specific queries.
- Electronic Commerce. Commerce is almost entirely driven by human interactions; when to buy goods, how much to pay, and so on. A simple electronic marketplace creates "buying" and "selling" agents for each "item" to be purchased or sold respectively. Commercial transactions are made by the interactions of these agents.
- Business Process Management. Company managers make informed decisions based on a combination of judgement and information from many departments. The idea is to see a business process as a community of negotiating, service providing agents. Each agent represents a distinct role or department in the enterprise capable of providing one or more services. This agent-based approach means that services can be scheduled in less time and service exceptions can be detected and handled.

2.3.3.3 Entertainment

Agents play an obvious role in computer games, interactive theater, and related virtual reality applications: such systems tend to be full of semi-autonomous animated characters, which can naturally be implemented as agents. The agents are programmed in terms of behaviors. Which are simple structures, which resemble rules but do not require complex symbolic reasoning. Interactive theater is a system that allows a user to play out a role analogous to the roles played by human actors in films.

2.4 Electrical Power Distribution Systems

An Electrical Power System is composed by a group of one or more generating resources and connecting transmission lines operated under common supervision to supply consumers [16]. A power-delivery system is everything that exists between power generation (e.g. generators, batteries, etc.) and the specific consumer of power (e.g. computers, motors, weapon systems, etc.) [16]. It is constituted by:

- Transmission system. Transports high voltage electrical energy over long distances. This high voltage electricity is reduced at a major load center and is then sent to the final users.
- Distribution system. The Distribution System transports electrical energy from the transmission system to the final user.

Electric power distribution systems can be found almost everywhere, from ship power systems to data centers. The EPDS used in this study are self-contained EPDS [5]with zonal distribution. Next, we will summarize the principal definitions and concepts related to zonal ship design and zonal architectures; also, we will introduce IPS and DCZEDS.

[5] Examples of self-contained power systems are the power systems in automobiles, aircraft, and manufacturing plants. They have been commonly used for testing reconfiguration.

2.4.1 Zonal Ship Design

Capt. Norbert Doerry, technical director for Future Concepts and Surface Ship Design at the Naval Sea Systems Command of the US Navy presents a review of the state of the art zonal ship design in [17]. He also proposes a framework for zonal ship design to satisfy survivability and quality of service (QOS).

Survivability is defined as "the ability of the distribute system, even when potentially damaged by a threat, to support the ship's ability to continue fulfilling its missions to the degree for the particular threat" [17]; it measures the capability of the ship to continue operating under a fault condition. On the other hand, QOS "measures the ability of the distributed system to support the normal, undamaged operation of its loads" [17]. The best configuration of a distributed system may be different for survivability considerations and for QOS considerations [17].

In order to talk about Zonal Ship Design we need to clarify some key terms [17]:

- Zone. "A zone is a geographic region of ship" [17]. The dimensions and boundaries of a zone are defined in order to maximize survivability.
- Adjacent Zones. If two or more zones could concurrently be damaged by a design threat, they are call adjacent zones.
- Zonal Survivability. Is the ability of a zonal system to support the normal operation in undamaged zones.

In a zonal architecture the ship is divided into a number of electrical zones that are delineated by physical watertight bulkhead compartments [18].

Zonal architectures can be classified according to the quantity of buses: Single Bus Architectures, Dual Bus Architectures and Multiple Bus Architectures. Hybrid Bus Archi-

tectures combine a single bus with two buses. A complete description of these architectures and the variations of each type can be found in [17].

Zonal architectures have several advantages over traditional ring bus distribution systems including better reconfigurability, the supply of radial feeders, and greater survivability. They provide fault tolerance, reduced cabling, and cost savings [17, 18, 19]. For our studies we used three simulation test systems, the first one based on a Single Bus Architecture and the others based on Dual Bus Architecture.

2.4.2 Ship Integrated Power System

The Naval Sea Systems Command Advanced Surface Machinery Program proposed a ship architecture called Integrated Power System (IPS) where a common electrical source supplies both, service loads and ship propulsion. A simulation testbed (based on the IPS) for control algorithms of ship power distribution system was provided by the Office of Naval Research (ONR) to researchers in the NSF/ONR Electric Power Networks Efficiency and Security (EPNES) program [20].

Figure 2.4 shows the IPS with power distribution based on the zonal distribution architecture, and it includes an AC backbone and a DC Zonal System [18, 19, 21, 22].

The system shown in Figure 2.4 contains the minimum elements required to represent an advanced IPS [19]. An actual ship would typically have five to eight zones instead of three [17, 19]. The model system characteristics include [19]:

1. Two finite inertia AC sources and buses.
2. AC bus dynamics, stability and regulation.
3. Redundant DC power supplies and zonal distribution buses.
4. DC bus dynamics, stability and regulation.

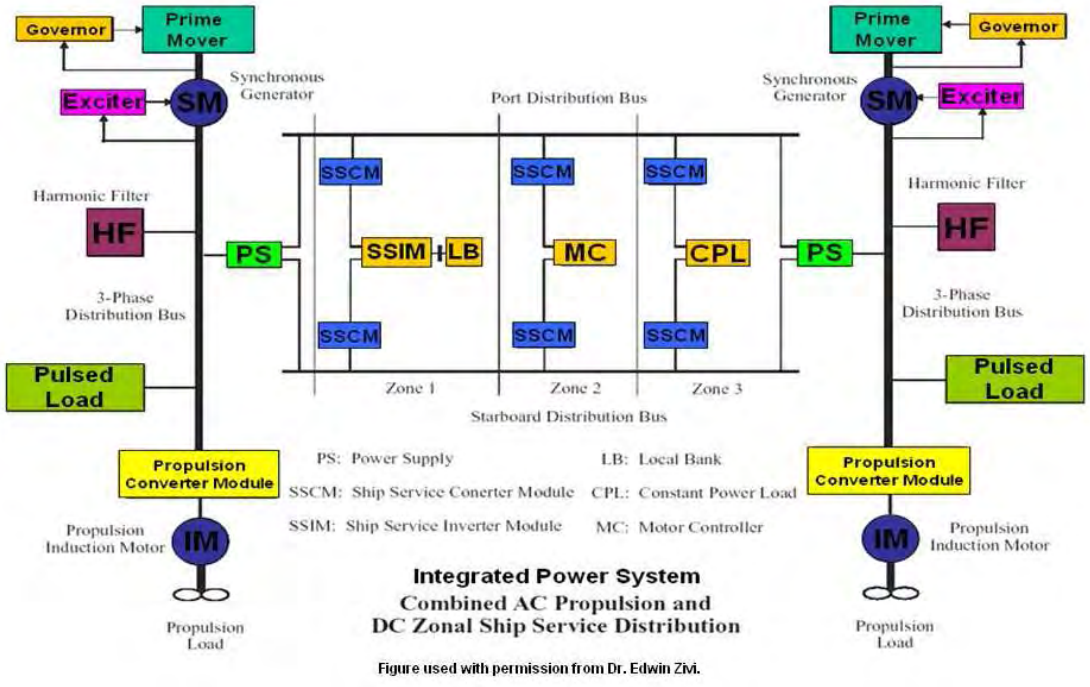


Figure 2.4: ONR Reference System: Notional Integrated Power System (IPS) [20, 19].

5. Three zonal distribution zones fed by redundant DC power buses.

6. A variety of dynamic and nonlinear loads.

2.4.3 DC Zonal Electric Distribution System

Figure 2.5 shows the DC-Zonal Electrical Distribution System (DCZEDS), which is the DC part of the IPS [18, 21]. The DCZEDS is based on a Dual Bus architecture [17].

As mentioned previously, the simulation testbed was provided by ONR to EPNES researchers. The reference DCZEDS is fed by two 500-V busses; one on the starboard side and one on the port side. A Ship Service Converter Module (SSCM) connects each bus to the zone; it serves for buffering the main bus and the zone electrical loads and it provides the appropriate voltage level to the load [18, 21, 22]. Diode networks are used for automatic bus transfer. AC loads are fed by the Ship Service Inverter Module (SSIM). The loads are divided into three zones as shown in Figure 2.5. All DCZEDS subsystems and modules

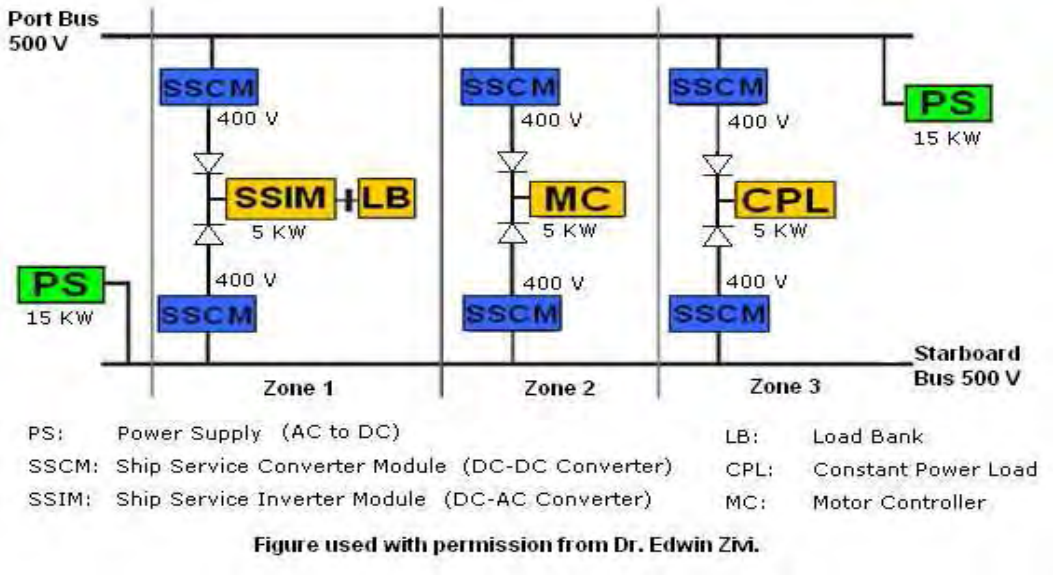


Figure 2.5: ONR IPS Reference System: DC Zonal Distribution System [19, 20].

have local controllers [23].

2.5 Reconfiguration and Survivability of EPDS

Reconfiguration and survivability are highly desirable capabilities in EPDS. A reconfiguration strategy identifies the current operating status of the controlled environment, it evaluates control performance and selects the appropriate system configuration. These actions are taken to reduce damage to the electrical system when faults occur and to prevent supply interruptions to vital loads. As previously mentioned, survivability of a system is its capability to fulfill a mission avoiding or withstanding a hostile environment caused by attacks, failures, or accidents [17, 23].

2.5.1 Self-Reconfigurable System Implementation Alternatives

Some research has been dedicated to the reconfiguration problem. Centralized and distributed approaches haven proposed to find a solution. Next, we will present some of the most important research carried out in the reconfiguration area.

Some of the most important research carried out in the reconfiguration area includes: Optimization and probabilistic methods, Genetic Algorithms, Artificial Neural Networks, Cellular Automata, State Machines and MultiAgent Systems. The first four propose solutions for the reconfiguration problem using a centralized approach and last two propose a distributed solution. Research dedicated to the study of MAS applied to reconfiguration will not be included here as it is presented in Section 2.6

2.5.1.1 Optimization and probabilistic methods

The reconfiguration method proposed by Davey and Hebner [24] considere the power system as a grid of interconnected trunk lines. The lines have equivalent parallel load and series transmission impedances, which are used as a tool to reduce a complex system into a compact system.

The reconfiguration algorithm tries to find the position of the switches maximizing power flow through the parallel impedances and minimizing losses through the series impedances; the decisions are subject to the constraint that the current passing through a line in steady state stays under its rating. The load flow is controlled by the state of the switches. The switches in the terminals of any trunk line cannot be opened or closed at the same time [24].

Srivastava and Butler-Puny [25] presented an automated probabilistic methodology for reconfiguration. The method calculates the probability of damage for each electrical component of the ship and also includes a heuristic method to determine reconfiguration control actions using the calculated probability. The analysis showed the effectiveness of the predictive reconfiguration method is dependent on the accuracy of the prediction of probability of damage. They concluded, too, that even for inaccurate predictions the results were encouraging.

Jin et al. [26] presented a method for load balancing, based on modified binary particle swarm optimization (BPSO), to find the optimal solution for a non-linear optimization reconfiguration problem. This method is subject to security and operational constraints. The authors concluded that by using BPSO the load balancing index is decreased by reconfiguration, which is an efficient solution.

2.5.1.2 Genetic Algorithms

Genetic algorithms (GAs) are a computer science technique used to approximate solutions to optimization and search problems. They are implemented as a computer simulation in which a population of abstract representations (called chromosomes) of candidate solutions (called individuals) to an optimization problem evolves toward better solutions [27].

Mendoza et al. [28] proposed a method that improves the adaptability and efficiency of GAs when they are applied to a minimal loss reconfiguration problem. They used a codification strategy and genetic operators (called accentuated crossover and directed mutation) to reduce the searching space (population). Their results showed a reduction in computational time and memory requirements. They also found better solutions than those achieved with non-modified GA or traditional algorithms.

2.5.1.3 Artificial Neural Networks

An artificial neural network (ANN) is an organized group of artificial neurons (simulations of biological neurons) that uses a mathematical or computational model for information processing.

Salazar et al. [29] proposed an ANN that determines the most suitable topologies to minimize the active power loss in the distribution system. They used clustering tech-

niques for the training sets, that result in more effective inputs for the ANN. Their results showed that the application of clustering techniques associated with validation techniques allows the construction of a reduced training set containing sufficient information to provide adequate learning by an NN. They presented that their NN performed equally as well as reconfiguration programs that employed traditional optimization approaches, and were superior to approaches based on NN algorithms for the same problem.

2.5.1.4 Cellular Automata

Cellular automata (CA) are a discrete model that consists of an infinite, regular grid of cells, each in one of a finite number of states. The cellular machine can show the following features: reconfigurability, upgradability, expandability, reusability and self-organization. Consequently, by introducing a self-organization algorithm to cellular machines, strategic decisions for the self-reconfigurable systems can be carried out under complex requirements [30].

Qiu-Xuan et al. [30] presented a method using genetic algorithms to find a configuration of cellular automata and rules within an enormous search space. Their results showed that the control algorithms using CA are efficient because they can find optimal, or near optimal solutions to the self-reconfiguration problem. The authors concluded that more concerted efforts and research in this direction is needed.

Even when the method presented in [30] is intended for controlling self-reconfiguring robots, it is a good approximation to the reconfiguration problem of EPDS and it presents a new path for research into this specific problem.

2.5.1.5 State Machines

Feliciano and Vélez [31] developed a real-time simulation framework of a shipboard integrated power system as a part of the IPRs project. The proposed framework is composed

of three parts:

- The DD(X) shipboard electric system. The DD(X) ship is the next generation surface combatant based on the IPS architecture proposed by the US Navy. This model represents the AC/DC combined distribution system, with components and parameters based on published data related to DD(X) ships.
- IPR hardware. Simulated using model blocks found in the SimPowerSystems and Simulink™ libraries.
- The control logic. Developed with state machines using the Stateflow™ toolbox.

They integrated the DD(X) shipboard electric system with the IPR model. Their results showed that the DD(X) was successfully reconfigured, in a distributed way and with acceptable recovery times for this type of reconfiguration scheme in the majority of the scenarios. The proposed model is qualified to work in AC systems. The authors suggest integrating the DC part in future research, which will require structural changes in the blocks used to simulate the IPR hardware model [31].

Notice that the authors centered their investigation in the development of the time-domain simulation framework of an IPR-based Shipboard IPS. The idea was to propose a tool that allows future research to explore more complex reconfiguration algorithms based on the IPR concept. Based on this goal, even when the proposed algorithm responded successfully to the evaluated contingencies, it is necessary to include some reconfiguration alternatives such as the capacity to respond when a fault is cleared, and the ability to select the lowest-cost path alternative to serve a load is required.

In addition, the state machines used by the reconfiguration controller presented in [31] are modular, but they are implemented in Matlab™ Simulink™ - Stateflow™ toolbox which is not a multiprocessor simulator. Although the state machines have concurrent behavior, they are running sequentially.

Our research aimed at implementing the reconfiguration logic using MAS in order to apply more complex algorithms and a negotiation process that allows us to respond to some of the challenges proposed by previous work. MAS are parallel and asynchronous. In addition, they are technologies, such as JAVA programming language and JADE platform, that allows the agent platform to be split on several host to implement a real distributed system.

2.6 MultiAgent Systems applied to System Reconfiguration

MultiAgent systems have been used in computer science and artificial intelligence applications and they are being studied as an approach to implement self-reconfigurable EPDS. Some research has been dedicated to study the application of MultiAgent systems to the problems of protection and reconfiguration of AC and DC power systems.

Tomita et al. [32] proposed a "relay agent" to build a cooperative protection system. The intelligent relays were distributed over the system and the agents, that move between equipment, carried out their protection functions by cooperating using a communication network. Their results showed that using relay agents the protection system could minimize the isolated zone for any changes in power system conditions and to secure high reliability with less redundant hardware. The authors recommended the optimization of the installation of relay agents into the equipment, from standpoints of communication amount, reliability and processing speed, for future research.

Thorp et al. [33, 34] centered their investigation on the use of agents within an electric power grid using a private communication network, in order to develop a system protection scheme. Their proposal uses geographically distributed agents located in a num-

ber of Intelligent Electronic Devices [6]. They also developed a simulation platform called EPOCHS. EPOCHS allows agents to operate in an environment that combines electric power, electromagnetic and electromechanical transient simulators with an event-driven network communication simulation engine.

Their agent-based protection system was tested using EPOCHS. The agents exchanged information between zones using the network communication and this increased protection of the system. The authors concluded that even when agents enhanced the performance of protection and control systems, they do not replace traditional protection systems. They also suggested the need to improve communication security and protocols for future work.

These two investigations intended to isolate the fault area and work with the protection system, but the reconfiguration problem is not addressed. They also used agents that are not FIPA-complaint.

Recent projects are starting to use FIPA specifications to develop their systems and to implement communication and negotiation strategies. Summary of some of these projects follows.

Sun and Cartes [35] have been working on the reconfiguration problem of a shipboard power system using agents. Each agent in the system only communicates with its immediate neighbors in order to reduce the dependency between the system topology and the reconfiguration algorithm. The authors assumed a radial topology to simplify the problem but they concluded that agents can act locally and with local information.

As a part of the previous research, an interface to communicate a MAS developed

[6] An Intelligent Electronic Device (IED) is a microprocessor-based controller of power system equipment, such as transformers, circuit breakers, and capacitor banks. http://en.wikipedia.org/wiki/Intelligent_electronic_device

in JADE with an electrical model simulated in Virtual Test Bed (VTB) was presented in [36]. This interface is a complex work that was available through the authors but was not documented at the time of our request. Therefore, we proposed a method to communicate JADE with Simulink^{TM}, which is a simple but useful tool for development and verification of control algorithms implemented with agents which will be explained in Chapter 4.

Both the interfaces allow the MAS developed in JADE to receive information about the status of connections and are limited to send signals to open/close switches in the EPDS. The are two main differences between the interface proposed by [36] and ours:

- The EPDS simulator used by [36] is VTB. We connected Simulink^{TM} with JADE.
- They used the Common Object Request Broker Architecture(CORBA). CORBA enables software components written in multiple computer languages to interoperate. Our interface uses the client/server model, connecting the EPDS and the MAS through TCP/IP sockets. A socket is one endpoint of a two-way communication link between two programs running on the network and bound to a port number [49].

Wang et al. [37] proposed a MultiAgent system to restore the power supply to as many loads as possible with higher priority loads restored first. The system is implemented in Virtual Test Bed (VTB) and Java Agent DEvelopment Framework (JADE). The MAS was tested on a small system that consisted of one facilitator agent and a group of bus agents.

Research [35] and [37] addressed some important requirements of a MAS-based EPDS but the priority scheme in these projects was restricted to vital and non-vital loads, and the use of FIPA specifications is limited to implementing the MAS with the JADE platform. We proposed to expand the priority scheme and the use of FIPA specifications to facilitate the interoperability and understanding of the MAS.

Researchers at Mississippi State University have been working in the reconfiguration problem using MAS and localized information [38]. The MAS is implemented using both Matlab™ and JADE to explore reconfiguration strategies, communication and data sharing, and the shipboard power system is simulated in VTB. The MAS implemented in Matlab™ was tested, some limitations of using Matlab™ as a tool to develop a MAS were found [7]. Due to the drawbacks of using Matlab™/Simulink™ to implement the MAS, the authors implemented the MAS in JADE and then integrated it with the simulation model in VTB.

The authors concluded that their work is still in development. The MAS have been tested in small electrical models. Their priority is to understand the interaction between agents in order to define the location of the agents, the amount of information needed, and the topology independence features.

We proposed an architecture for the system using agents based on the IPR concept; which proposes a suitable location for the agents. Also, we detail the principal processes in the MAS specifying each ACL message needed during the interaction of agents, which includes the information that each agent needs to send or receive in each step of the process. With this proposal we present an answer to some of the questions formulated in [38].

Maturana et al. [39] proposed an Agent-based Control system to provide control for auxiliary systems on capital ships. Their objective was to regulate the temperature of devices aboard capital ships using the fluid distribution system. The MAS is implemented according to FIPA specifications and uses Contract Net interaction protocol and market-based auction protocols. This research addresses some of the most important issues related to the reconfiguration problem; however, id does not reconfigure the electrical system, and the communication between agents does not emulate the physical connections, which is one

[7] More information about the experiments and the limitations of Matlab™ can be found in [38].

of the most important issues to the IPR concept and to reduce the dependency between the system topology and the reconfiguration algorithm.

2.7 Final Remarks.

This Chapter provided a background of the basic concepts used on this project and a survey of previous and related work. Much research has been dedicated to the reconfiguration problem in a centralized way and all reached good results. However, the drawbacks of traditional centralized control have been documented amply in the literature and now research efforts must focus on the distribute reconfiguration problem in order to respond to the challenge of new power systems.

Some technologies have been explored, looking for a distributed solution to the reconfiguration problem. One of these technologies is the MultiAgent Systems, which has been presented as a very suitable approach to develop distributed control of an EPDS. Based on previous work in this area and state-of-the-art technologies, we established a set of requirements for a MAS-based EPDS:

- To guarantee the interoperability of MAS, the design must be made based on FIPA-specifications.
- The MAS must use communication based on ACL Message and FIPA Interaction Protocols. For each interaction, it is necessary to define the information that each agent needs to send to the receiver.
- It is important to use technologies that allows the agent platform to be split on several host to implement a real distributed system.
- It is necessary to establish a set of configuration parameters that each agent must receive to know its relation with the EPDS.
- The agents must represent critical parts of the system, according to the architecture

of the EPDS.

- The communication channels between agents must emulate the physical connections of the EPDS.
- The loads on the EPDS can have different priorities, the MAS must act based on a priority scheme.
- When the MAS has more than one option to reconfigure the system, it is necessary to include a decision factor to choose the best option.

Some of these requirements have already been used in previous research. However, this thesis aims to answer some of the questions that were not addressed in previous work, therefore some other requirements were included in order to fill in this area. These questions have been formulated (but not answered) in the state-of-the-art and other research.

The next Chapter will present the design of the proposed MAS-based Electric Power Distribution System. This design is based on the IPR concept and the requirements presented above.

CHAPTER 3

Reconfiguration of the Zonal System using a MultiAgent System

3.1 Overview

This Chapter presents the aspects related to the design of the MAS-based Electric Power Distribution System. The next sections introduce first, the mathematical problem formulation, the general architecture of the system and the types of agents. The MAS is composed by two agents for each zone, ZoneAgent and CommunicationAgent. The ZoneAgent is the reconfiguration agent for the zone and the CommunicationAgent manages the communication between ZoneAgent and the EPDS.

Then, the scenarios in which the system operates will be presented based on the interaction between the agents. The main interaction in the system is the negotiation process. It will be explained along with the decision strategy used to fulfill the reconfiguration goal. At the end of the Chapter, we will give an example of the MAS operation.

3.2 Mathematical problem formulation for System Reconfiguration

Our approach is based on the idea that System Reconfiguration is an optimization problem, that has an objective function with a set of constrains. Our mathematical model is a modification of the formulations presented in [3] and [40]. The objective is to maximize the number of served loads with highest priority. Therefore, we need to establish an objective function where the contribution of high priority loads is greater than the contribution of low priority loads. The loads are multiplied by a weighting factor $W_i = (\alpha - P_i)$, where $P_i \in [1, M]$ is the load priority (the highest priority is 1 and the lowest M.), and α is an integer larger than M.

The following mathematical expression represents the main function:

$$Max \sum_{i \in L} y_i \times W_i \times L_i$$

Subject to:

Equality Constraints

$$\sum_i I_{in_i} = \sum_i I_{out_i} + L_i \times y_i$$

Line Limits

$$I_{i\_j} \leq I_{i\_j}^{\max}$$

Voltage Limits

$$V_i^{\min} \leq V_i \leq V_i^{\max}$$

$$i \in FN, \wedge, j \in TN$$

Where,

- y_i is a binary variable ($y_i = 1$ if the load is served and $y_i = 0$ if the load is disconnected).
- V_i is the voltage of bus i
- L_i is each load in the system i
- $I_{in_i}$and $I_{out_i}$are the currents entering and leaving bus i
- FN is a set of "from buses"
- TN is a set of "to buses"

3.3 A MultiAgent System Architecture for the Reconfiguration of the Zonal System

The advantages of zonal architectures and the fundamental role that they are playing in the present and future of electric distribution systems were mentioned in Chapter 2. Therefore, we selected a zonal distribution in order to design the MAS-based EPDS. Figure 3.1 presents the general vision of an IPRs-based zonal system where each zone has a "Zonal IPR". Each IPR has agents with embedded intelligence to reconfigure the system.

Figure 3.2 shows the proposed MAS architecture. The Agent platform is composed of several agent containers; a container is a place where the agents live. The main container has the agents required by FIPA specification: Agent Management System (AMS) and Agent Directory Facilitator (DF). Also, the platform must contain a Message Transport Service (MTS).

The agent platform also has one container for each Zonal IPR with two types of agents, ZoneAgent and CommunicationAgent. The first one represents the zone and the other provides a communication bridge between ZoneAgent and the EPDS. Each ZoneAgent

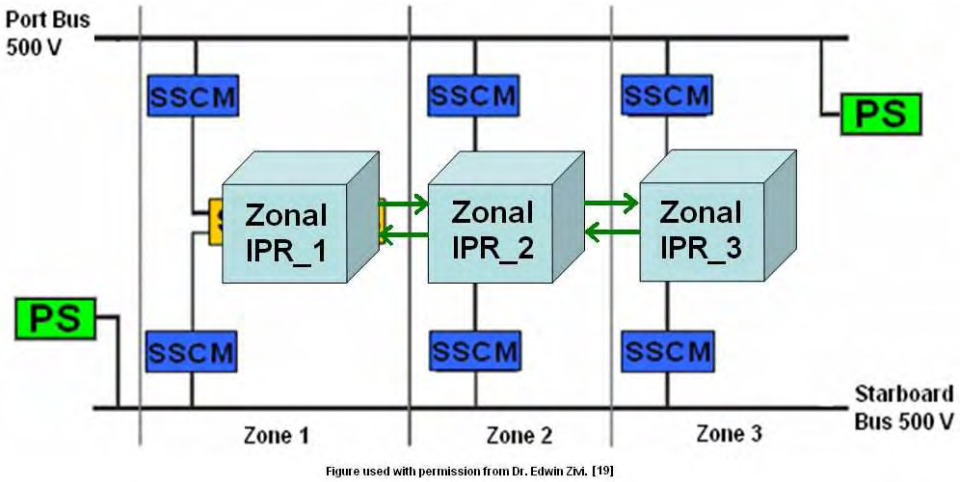


Figure 3.1: IPRs-based zonal system.

knows the state of its zone at all times and it only can interact with its immediate neighbors, emulating the physical zone connections in the EPDS.

3.4 Types of Agents in the MAS

As previously mentioned, the agent has two types of agents for each zone, ZoneAgent and CommunicationAgent.

Notice that the MAS design is modular. Each zone has exactly the same agents and the difference between them is given by the customization of a set configuration parameters at the beginning of the simulation. At the beginning of their life cycle, each agent acquires the configuration parameters from a plain file and initializes the variables and behaviors based on these parameters. For each zone it is necessary to instantiate exactly the same agents.

3.4.1 ZoneAgent

The ZoneAgent represents the zone and negotiates with its immediate neighbors to reconfigure the system in the event of a fault. The principal goal of the ZoneAgent during

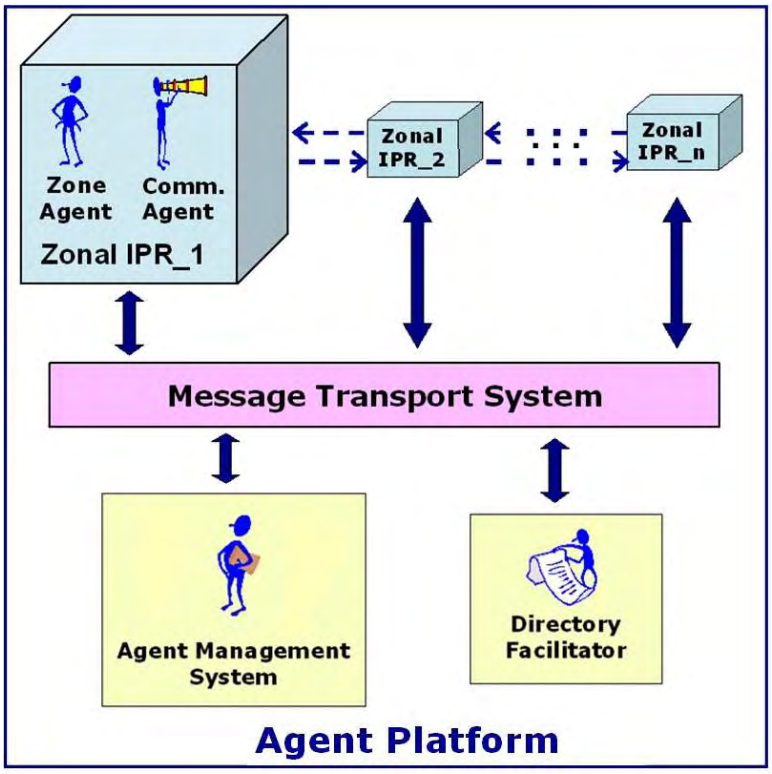


Figure 3.2: MAS-based proposed Architecture.

reconfiguration is to maximize the number of served loads with highest priority. Also, the ZoneAgent may have more than one option to serve each load and it must select one of them.

The principal aspects of operation of a ZoneAgent are summarized next:

- ZoneAgent needs to know the information about the priority of the zone, the names of the neighbor agents, the power requested by its load, the number of buses, and the generation capacity per bus. At the beginning of its life cycle it acquires this information and initializes the variables and behaviors based on these parameters.
- Each zone can have one or more buses with generation capacity. The ZoneAgent, based on negotiation and a priority scheme, distributes the power available. It must store the information of the zones that it is serving, including its own zone (if applicable).

These zones are called “Clients”.

- Each zone can have a load. The ZoneAgent must select the source or sources needed to serve its load. The sources are stored as “Helpers”. They are selected based on the maximization of the loads served with the highest priority. If the ZoneAgent has more than one option, it uses the cost-path of each proposal to make a decision. Its own source bus-connections have different costs, the first bus has the lowest cost and the last bus has the highest cost. The use and calculation of the cost-path will be explained in Section 3.6.3

- When a ZoneAgent cannot serve a client, it can redirect the request (or a part) to a neighbor ZoneAgent. If a connection between no-neighbors zones is required the ZoneAgent must use its zone to act as a bridge in the reconfiguration process.

  For example, ZoneAgent1 asks to ZoneAgent2 for help, but this zone cannot help and the message it is redirected by ZoneAgent2 to ZoneAgent3. ZoneAgent3 helps and stores ZoneAgent2 as a client. ZoneAgent1 has ZoneAgent2 as a helper. ZoneAgent2 knows that its zone is a bridge and stores ZoneAgent1 as a client and ZoneAgent3 as a subcontractor. In this case, if ZoneAgent2 has some power available but not enough, the proposal can use power from Zone2 and Zone3.

- With the information of clients, helpers, and bridges, the ZoneAgent must establish the control actions to reconfigure the system. These actions must be sent to the CommunicationAgent. The control actions are orders to open/close switches in the EPDS. These switches connect the load with the zone, interconnect the zones, and connect each SSCM with its respective bus.

3.4.2 CommunicationAgent

The CommunicationAgent provides a communication interface between the ZoneAgent and the EPDS. The principal aspects of operation of a CommunicationAgent are summa-

rized next:

- CommunicationAgent needs to know general information about the zone, such as number of buses and the name of the ZoneAgent. At the beginning of its life cycle it acquires all this information and initializes the variables and behaviors based on these parameters. Also, it establishes a communication channel with its ZoneAgent.
- The CommunicationAgent interacts with the EPDS and establishes a way to receive and to send information from/to the EPDS. The best way to implement these characteristics must be determined according to the specifications of the EPDS. In Chapter four of this document a simulation implementation of a MAS-based EPDS is presented.
- The CommunicationAgent receives from the EPDS the information about the status of the connection to each bus and it informs the ZoneAgent when a change in the system conditions occurs.
- The CommunicationAgent receives from the ZoneAgent the control actions to reconfigure the system and it sends them into the EPDS.
- When the control actions are sent, the CommunicationAgent waits for confirmation from the EPDS. If the verification is unsuccessful, the agent must re-sent the control actions to the EPDS. This verification process includes a deadline by which confirmation should be received by the CommunicationAgent or the process is declared unsuccessful.

3.5 Interaction between Agents to reconfigure an EPDS

This section explores general processes of the MAS-based EPDS and the messages used. The main interaction of the MAS-based EPDS is the Negotiation Process which is presented in Section 3.6.

It is important to know that even when there are sequential processes, for the

asynchronous nature of the agents, one or more could be running in parallel at the same time, using different behaviors.

3.5.1 Initialization

At the beginning of their life cycle, the agents acquire all the information about the system and the zone, that it is relevant for them. Also, each CommunicationAgent sends a SUBSCRIBE message to its ZoneAgent for establishing a communication channel. The general structure of the SUBSCRIBE message used in this process is shown next:

```
(subscribe
    :sender CommunicationAgent
    :receiver ZoneAgent
    :content (CommunicationAgent information)
    :language fipa-sl
)
```

The content of the messages includes the identification information of CommunicationAgent, which is used for authentication purposes.

3.5.2 Fault Condition

When a fault situation occurs the CommunicationAgent sends a REQUEST message to report the new status of the bus-connection and to ask for control actions. The general structure of REQUEST message is shown next:

```
(request
    :sender CommunicationAgent
    :receiver ZoneAgent
    :content (information of zone bus-connections status)
    :language fipa-sl
)
```

When the ZoneAgent receives the REQUEST message, it starts a process in order to reconfigure the system. A flowchart of the process is shown in Figure 3.3.

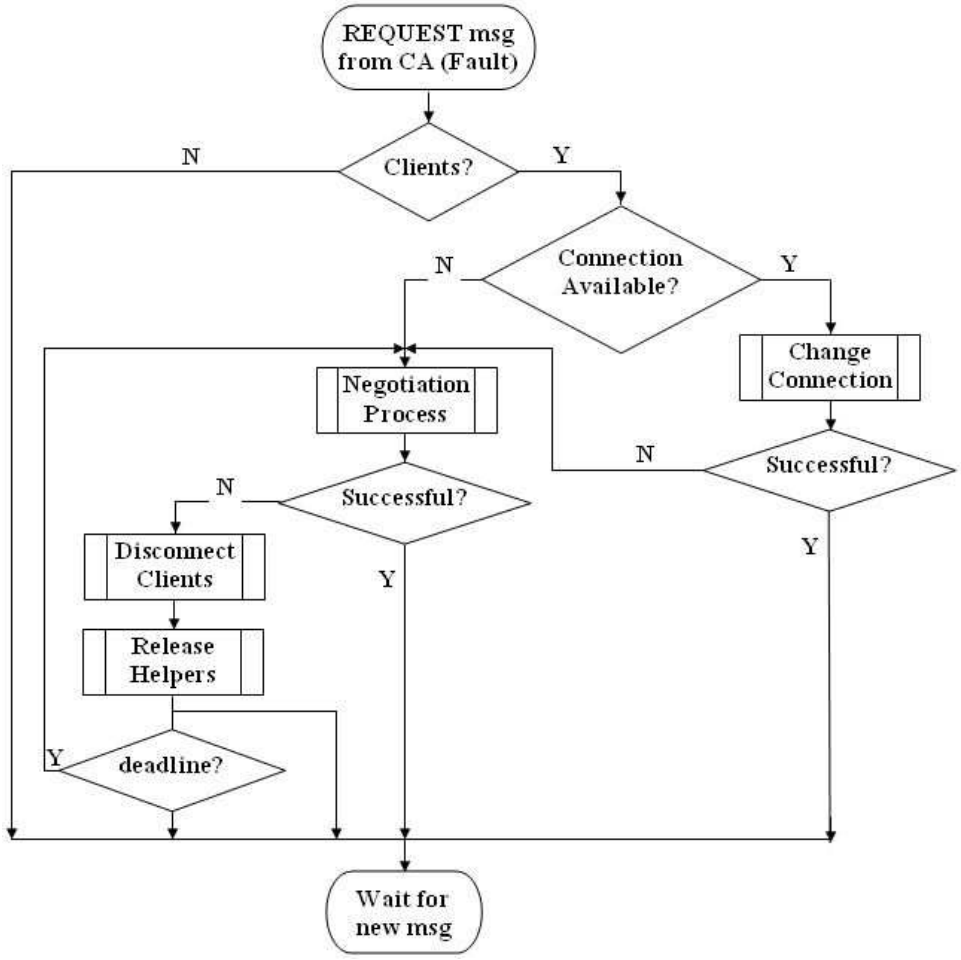


Figure 3.3: Flow Chart of "Fault Condition" process.

The process consists of actions that the ZoneAgent must take according to the information and the function it is performing. The ZoneAgent executes all of these actions for each client of the zone, including its own load. The details of these actions and the most important messages used are presented next.

3.5.2.1 Change Connection

If a helper ZoneAgent, with a fault situation, can serve its clients using an alternative bus-connection, it starts an interaction process to inform of the new situation and to coordinate the change. If the zones are connected using a bridge zone, the helper

ZoneAgent starts the process with the bridge ZoneAgent, who interacts with the next bridge ZoneAgent until the process reaches the client ZoneAgent. The sequence diagram of this process is shown in Figure 3.4.

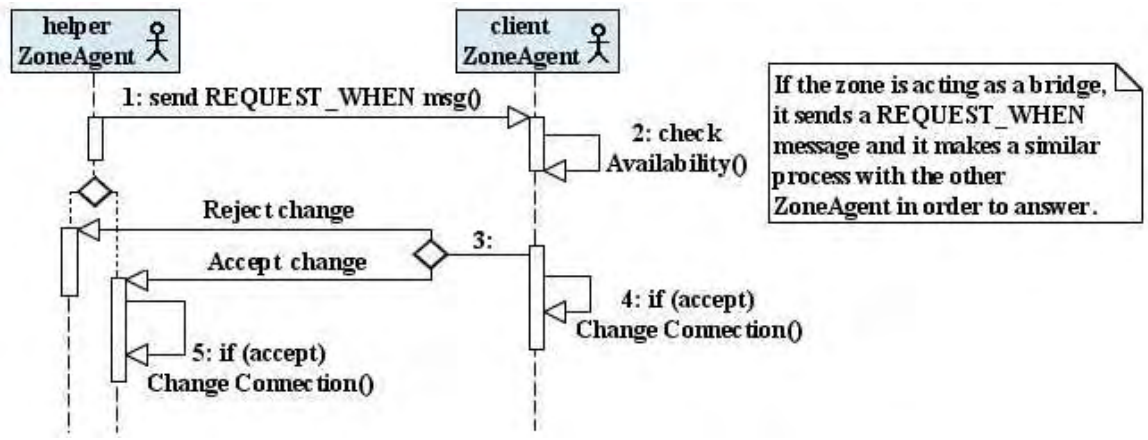


Figure 3.4: Sequence Diagram of Change a Bus-Connection.

The helper ZoneAgent sends a REQUEST_WHEN message to ask for a disconnection in the current bus and a reconfiguration that allows the system to use the requested bus-connection. The structure of REQUEST_WHEN message is:

```
(request_when
    :sender helper_ZoneAgent
    :receiver client_ZoneAgent
    :content (proposal id, bus requested)
    :language fipa-sl
)
```

The content of the message includes the proposal "id" the ZoneAgent is asking to move and the requested bus. The proposal id is used as a receipt or contract number between the ZoneAgents. The client ZoneAgent receives the message and it can accept or reject the new connection.

In special cases when the client is the same helper, no messages are passed and the

ZoneAgent can change the connection automatically.

3.5.2.2 Disconnect Clients

When a zone can no longer help, the helper ZoneAgent sends a DISCONFIRM message to the client ZoneAgent. The structure of this message is shown next.

```
(disconfirm
    :sender helper_ZoneAgent
    :receiver client_ZoneAgent
    :content (proposal id, power revoked)
    :language fipa-sl
)
```

The content of the message includes the proposal "id" and the amount of power that cannot be served.

When a ZoneAgent receives a DISCONFIRM message, the situation is approached as a "Fault Condition" (see Figure 3.3) but it only treats the client associated with the proposal. This client could be the zone itself or a neighbor zone if the ZoneAgent is acting as a bridge.

3.5.2.3 Release Helpers

When help is no longer required, the client ZoneAgent sends a CANCEL message to each helper ZoneAgent. The general structure of this message:

```
(cancel
    :sender client_ZoneAgent
    :receiver helper_ZoneAgent
    :content (proposal id, power returned)
    :language fipa-sl
)
```

The content of the message includes the identification of the proposal. Also, it includes the power returned, because a ZoneAgent can cancel only part of the power that is being served. When a ZoneAgent receives a CANCEL message and it is acting as a bridge, it sends a CANCEL message to all helper subcontractor ZoneAgents associated with the proposal.

After an unsuccessful negotiation process, the ZoneAgent must disconnect its clients and release its helpers. Then, the ZoneAgent waits some time and starts a new negotiation process. Because of the parallel process of the agents, the status of the system can change. If the load is connected the timer is set to zero and the ZoneAgent cancels the delay.

3.5.3 Fault is cleared

When a fault is cleared, the CommunicationAgent sends a REQUEST message to report the new status of the bus-connection and to request control actions.

When the ZoneAgent receives the REQUEST message, it starts a process to reconfigure the system. A flowchart of this process is shown in Figure 3.5.

If the zone has the load disconnected, the ZoneAgent connects the load. If the load is connected but the cost-path of the bus-connection using the new resources is lower than the current load, the ZoneAgent changes the bus-connection and releases the helpers associated to the proposal (if any). Some of the methods used have already been explained, other methods are presented next.

3.5.3.1 Connect Load

This method does not require interaction with other ZoneAgents. The ZoneAgent stores the information of its own zone as a client and sets all variables associated with this

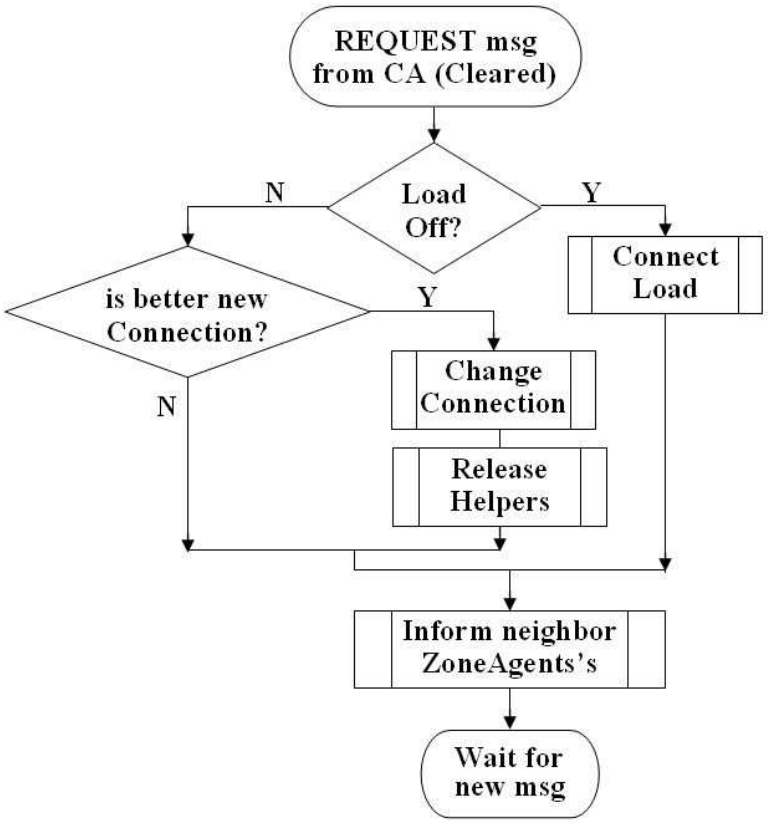


Figure 3.5: Flow Chart of "Fault is cleared" process.

action.

3.5.3.2 Inform Possible Clients

When a zone has new resources available, the ZoneAgent sends an INFORM_IF message to inform other ZoneAgents about the availability of resources. This is an invitation to start a negotiation process if they need it.

```
(inform_if
    :sender ZoneAgent_A
    :receiver ZoneAgent_B
    :language fipa-sl
)
```

When an agent receives an INFORM_IF message it spreads the information by sending it to its neighbor ZoneAgents.

3.5.4 Apply control actions

When the ZoneAgent needs to apply control actions on the EPDS, it sends a REQUEST message to the CommunicationAgent with the information. The general structure of REQUEST message is:

```
(request
    :sender ZoneAgent
    :receiver CommunicationAgent
    :content (information of control actions)
    :language fipa-sl
)
```

3.6 Negotiation for Reconfiguration and Control Actions

As mentioned in Chapter 2, FIPA has standardized, as protocols, some of the interactions between agents [14].

The negotiation process is the main interaction during the reconfiguration process in the MAS, for that reason we decided to use one of the pre-agreed message exchange protocols defined by FIPA.

We needed a negotiation process in which a ZoneAgent could ask other ZoneAgents to reconfigure the system, in order to maximize the number of loads served with the highest priority. Based on the requirements of the process, we needed an auction or negotiation protocol. FIPA standardizes some of these interaction protocols [14] and we decided to use the Contract Net Interaction Protocol.

In the following sections, we present the FIPA Contract Net Interaction Protocol

and its application to the negotiation process in the MAS.

3.6.1 FIPA Contract-Net Interaction Protocol

The Contract-Net Protocol was originally developed by Smith and Davis [41]. FIPA Contract-Net is a modification of the original Contract-Net protocol, it adds rejection and confirmation communicative acts. The representation of the FIPA-Contract-Net is shown in Figure 3.6.

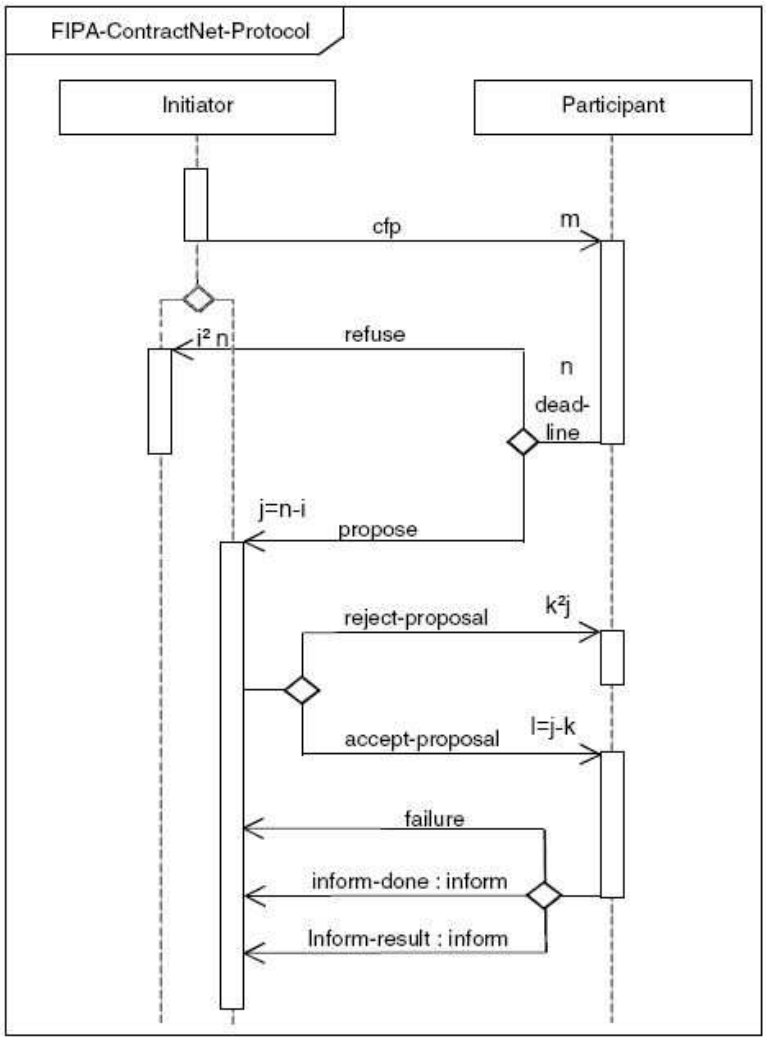


Figure 3.6: FIPA Contract Net Interaction Protocol [42].

In the Contract-Net Interaction Protocol, one agent (the Initiator) wishes to have some task performed by one or more agents (the Responders) and it establishes the preferences of the negotiation. For a given request, the Participants may send a proposal or they may refuse. Then, the negotiation continues with the Participants who send a proposal. Once the initiator receives the proposals, they are evaluated, the best is/are selected and it informs the agent(s) that will perform the task with an acceptance message (those not selected will receive a notice of rejection). Once the initiator accepts the proposal, the responder commits to perform the task. Once the responder has completed the task, it sends a completion message to the initiator [42].

The protocol requires that the initiator know when it has received all the replies, on the initiator may potentially be left waiting indefinitely. For this reason, the "call for proposals" includes a deadline by which replies should be received by the initiator. Proposals received after the deadline are automatically rejected [42].

3.6.2 Messages used in the Negotiation Process

FIPA establishes a set of pre-agreed messages for the Contract Net Interaction Protocol. The structure of each message is presented next, along with the message used in the negotiation process of our MAS.

- ACL Message CFP.

  This message is sent by the Initiator to ask for resources only a ZoneAgent can send CFP to its immediate neighbor ZoneAgents. The structure of the messages is:

```
(cfp
    :sender ZoneAgent_Initiator
    :receiver ZoneAgent_Responder
    :content (priority, code of negotiation,
            power requested per bus, specific bus)
    :protocol FIPA-Contract-Net
    :language fipa-sl
)
```

The content includes:

- The priority of the zone that needs the resources.
- The "code of negotiation" used to inform the responder if the message is sent by a bridge zone.
- The power requested per bus. If the zone has different resources available in each bus-connection, it can ask for a varying amounts of power to complete the required by the load.
- If a zone can only connect its zone to a specific bus, the ZoneAgent includes this parameter in the message.

- ACL Message REFUSE.

  A refuse message is sent by the Responder when it cannot send a proposal. The structure of the message is shown next:

```
(refuse
    :sender ZoneAgent_Responder
    :receiver ZoneAgent_Initiator
    :protocol FIPA-Contract-Net
    :language fipa-sl
)
```

- ACL Message PROPOSE.

  This message containing the information of the proposal is sent by the Responder. The structure of the message is:

```
(propose
    :sender ZoneAgent_Responder
    :receiver ZoneAgent_Initiator
    :content (per bus
                (power offered, cost, load shedding,
                power offered without load shedding)
            )
    :protocol FIPA-Contract-Net
    :language fipa-sl
)
```

The following information is include for each bus-connection the ZoneAgent is making a proposal:

- Power offered. The resources the zone can provide according to the priority scheme. For instance, if the zone has a client with priority 3, it can disconnect this client to offer resources to a zone with priority 1.
- Cost. The cost path of the proposal.
- Load shedding. A boolean operator that indicates if the ZoneAgent has disconnect a client in order to provide the power offered.
- Power offered without load shedding. The resources the zone can offer without disconnecting a current client.

- ACL Message REJECT_PROPOSAL.

  The initiator evaluates the proposals. If a proposal is not selected, ZoneAgent sends the following REJECT_PROPOSAL message to the Responder.

```
(reject_proposal
    :sender ZoneAgent_Initiator
    :receiver ZoneAgent_Responder
    :protocol FIPA-Contract-Net
    :language fipa-sl
)
```

- ACL Message ACCEPT_PROPOSAL.

  When the initiator evaluates the proposals, it sends the following ACCEPT_PROPOSAL message to each responder with an accepted proposal.

```
(accept_proposal
    :sender ZoneAgent_Initiator
    :receiver ZoneAgent_Responder
    :content (power accepted, bus selected)
    :protocol FIPA-Contract-Net
    :language fipa-sl
)
```

  The content of the message includes the amount of power accepted and the bus-connection that was selected.

- ACL Message FAILURE.

  When a Responder with an accepted proposal cannot perform the requested action, it sends the following FAILURE message to the Initiator.

```
(failure
    :sender ZoneAgent_Responder
    :receiver ZoneAgent_Initiator
    :protocol FIPA-Contract-Net
    :language fipa-sl
)
```

  A failure in the negotiation process can occur for different reasons, such as: a request for help of a higher priority zone received during the negotiation, a fault in the proposing zone, communication problems between agents, among others.

- ACL Message INFORM.

  A Responder with an accepted proposal sends an INFORM message onceit successfully performes the requested action.

```
(inform
    :sender ZoneAgent_Responder
    :receiver ZoneAgent_Initiator
    :protocol FIPA-Contract-Net
    :language fipa-sl
)
```

We included two additional messages, CONFIRM and PROPAGATE. These messages are not part of the Contract Net Protocol but they are required during the negotiation process. The explanation and structure of each follows.

- ACL Message CONFIRM.

  If a Responder needs to subcontract part of the resources to another zone (it acts as a bridge), it sends an INFORM message when it accepts the proposal. Only once the initiator accepts the proposal that the bridge sends, can the bridge ZoneAgent confirm the requested action to the subcontractor ZoneAgent.

For instance, ZoneAgent1 sends a CFP to ZoneAgent2. ZoneAgent2 does not have enough resources and starts a negotiation process with ZoneAgent3. ZoneAgent2 accepts the proposal of ZoneAgent3 but, ZoneAgent3 knows that ZoneAgent2 is acting as a bridge and it does not perform the requested action. When ZoneAgent1 accepts the proposal of ZoneAgent2, ZoneAgent2 sends a CONFIRM message to ZoneAgent3 and both perform the requested action.

The structure of a confirm message is shown next:

```
(confirm
    :sender ZoneAgent_Responder_asInitiator
    :receiver ZoneAgent_Responder
    :language fipa-sl
)
```

- ACL Message PROPAGATE.

  When an agent is acting as a bridge, it is managing resources of other zones. If a zone with a higher priority than a zone client needs some of those resources, the ZoneAgent sends a DISCONFIRM message to the client zone and a PROPAGATE message to the subcontractor helper zone. The structure of PROPAGATE message is:

```
(propagate
    :sender ZoneAgent_A
    :receiver ZoneAgent_B
    :content (proposal id, priority client)
    :language fipa-sl
)
```

  The content of the message has the id of the proposal that is being redirected and the priority of the new client.

3.6.3 Basic Negotiation and Decision Scheme

The main operation of the MAS negotiation process can be summarize as follows:

- When a ZoneAgent receives a CFP, it tries to build a proposal with the lowest cost possible. First, it verifies if the zone has enough power available to help its neighbor

zone. If the zone does not have enough resources, it initiates an inner negotiation process with its other neighbor ZoneAgents to built a compound proposal.

If it is not possible to send a proposal without load shedding, the ZoneAgent must verify the priority of the client zone, current clients and proposers of the inner negotiation process. The ZoneAgent tries to build a proposal using the available resources (of its zone and proposers) and full resources of current clients and proposers with lower priority than the client zone. Once a proposal is built, the ZoneAgent sends it to the initiator ZoneAgent.

- The initiator can receive more than one proposal, and each proposal can contain information from more than one bus-connection. Then, the ZoneAgent needs to find the lowest-cost option that maximizes the load served according to the priority scheme.

  The ZoneAgent must first check the resources offered without load shedding. If this option is not possible, the resources of current clients and responders are considered based on the priority scheme. Only zones with priority lower than the initiator zone would disconnect their loads.

  This process is carried out by each requested bus-connection, but the load can only be served by one bus-connection, so the agent must select the best option. The ZoneAgent uses the total cost for the proposal or combination of proposals to make the decision.

During the explanation of the negotiation process presented above, we introduced the decision factors that the ZoneAgent is using to make the decisions for the reconfiguration of the system. These two factors are Load Priority and Cost.

- Load Priority Factor. Each load has a priority factor. The loads with high priority are considered for reconfiguration before the lower priority loads.

  For example, a load with priority 3 is less important for reconfiguration purposes than a load with priority 1. If a zone only has resources to feed its load with priority 3, but

the ZoneAgent receives a request to feed a load with priority 1; the ZoneAgent of the zone with priority 3 must disconnect its own load to feed the higher priority zone.

- Cost Factor. The total cost of a proposal is composed of cost-path and load-shedding-priority cost.

$$total\_cost = cost\_path + load\_shedding\_cost$$

To explain the calculation of the cost we will use a graph representation of an EPDS with the associated cost of each edge. As an example of an EPDS we will use the Basic Double Bus DC Zonal System Model (one of the test systems used in this study). Figure 3.7**a** shows the schematic of the EPDS, and Figure 3.7**b** depicts a graph modeling the EPDS.

As an example, we assume that we have two connections available: the connection of Zone2 to the Port Distribution Bus and the connection of Zone3 to the Starboard Distribution Bus. The priority scheme is: Zone 1 has the highest priority 1, Zone 2 has priority 2, and Zone 3 the lowest priority 3. The analysis is made by ZoneAgent1. The costs were assigned based on the normal operation of the system.

  - Cost-path between the zone client and the zone(s) helper(s). The cost-path is the sum of the costs of all edges that form the path.

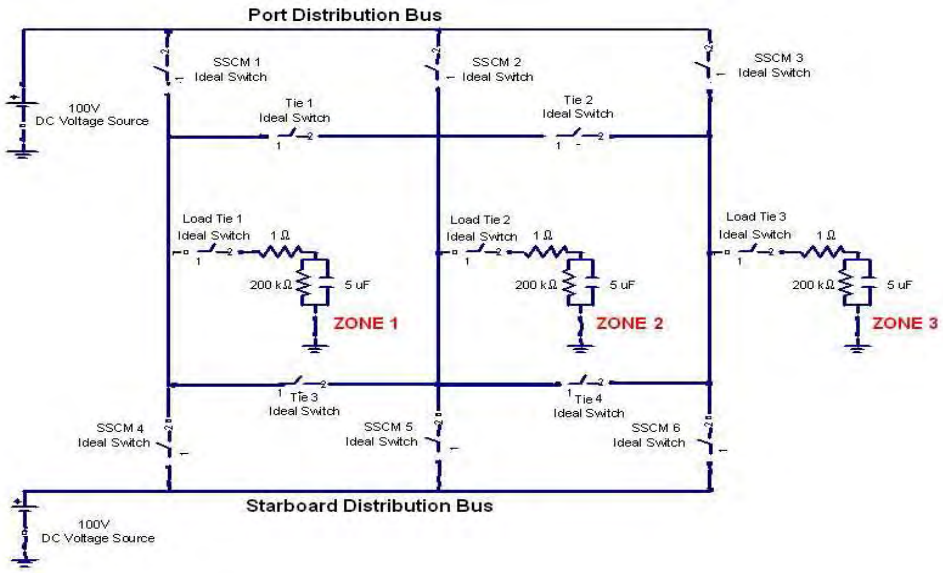
$$cost\_path = \sum_{p \in P} cost(p)$$

Zones have a different value for each bus-connection, as it is shown in Figure 3.7**b**. In the particular example, the load of Zone1 has two possible options: use the connection of Zone2 or the connection of Zone3. The paths are shown in Figure 3.8 and the cost_path are:

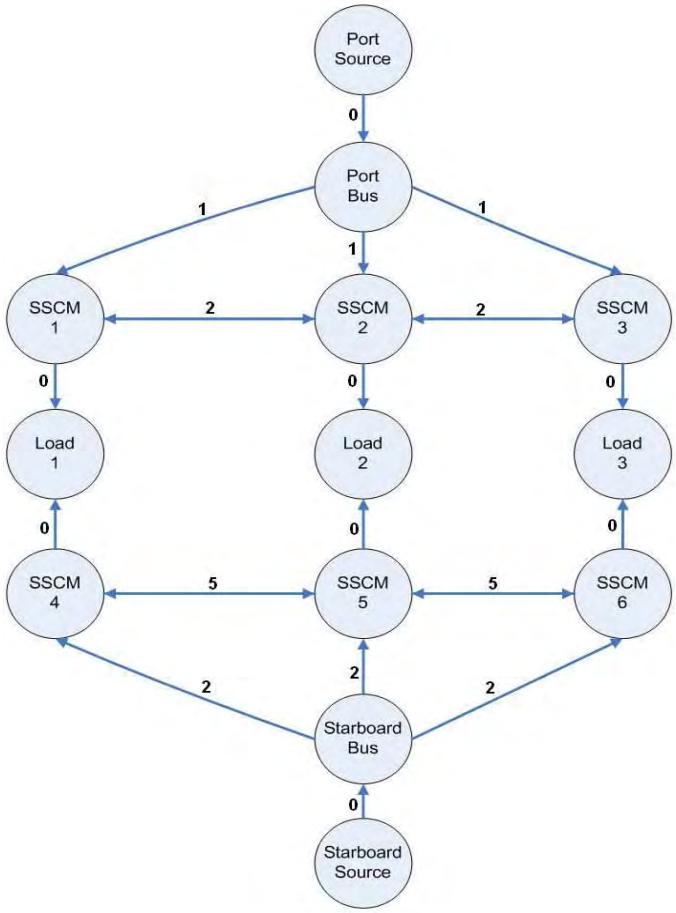
$$cost\_path(Zone2) = 0 + 1 + 2 + 0 = 3$$

$$cost\_path(Zone3) = 0 + 2 + 5 + 5 + 0 = 12$$

  - Load-shedding-priority cost. As previously mentioned, the first goal of a ZoneAgent is to maximize the number of loads served with highest priority. For this reason



(a) Basic Dual Bus DC Zonal System Model Schematic.



(b) EPDS Graph representation.

Figure 3.7: Modeling an EPDS as a graph.

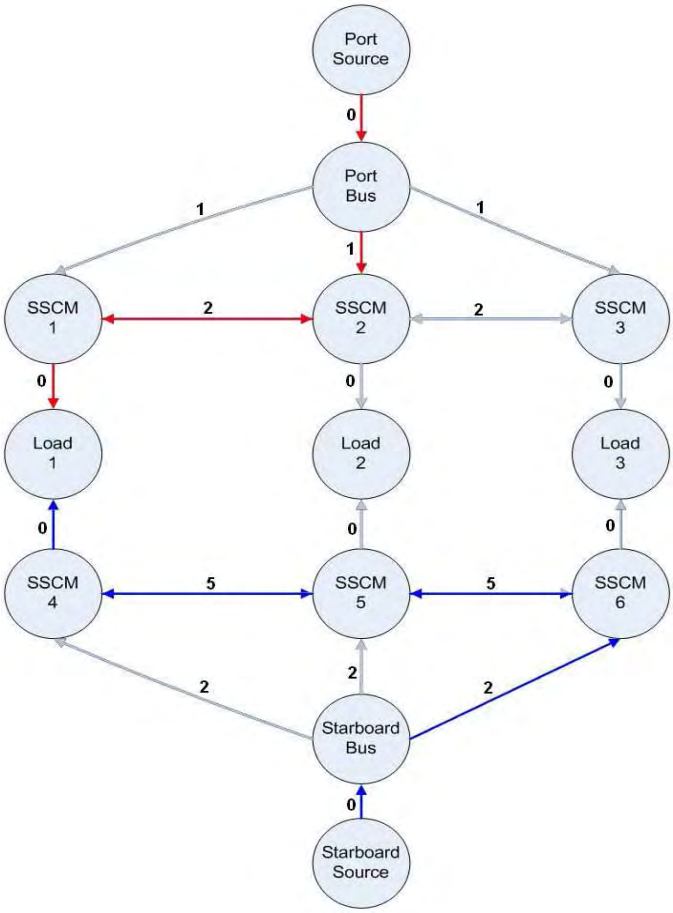


Figure 3.8: EPDS Graph representation - Path example.

it tries to find options to connect its load or to serve the load of a neighbor without disconnecting loads in the system. When the ZoneAgent has some options (proposals or combination of proposals) it must evaluate them based on more than cost-path. Therefore, a cost directly proportional to the priority is added to the cost of each proposal. This is the cost to the system for the disconnection of the load(s).

$$load\_shedding\_cost = \sum_{l \in loads2shed} \frac{1}{priority(l)} \times cost\_shed$$

The load-shedding-priority cost is used in the final step of the decision scheme, when only the proposals with lower priority than the initiator agent are being considered and the options to reconfigure the system without them have been

explored unsuccessfully.

In the example:

$$load\_shedding\_cost(Zone2) = \frac{1}{2} \times 90 = 45$$

$$load\_shedding\_cost(Zone3) = \frac{1}{3} \times 90 = 30$$

For the example presented the total cost of each bus-connection-proposal is:

$$total\_cost(Zone2) = 3 + 45 = 48$$

$$total\_cost(Zone3) = 12 + 30 = 42$$

Based on that results ZoneAgent1 selects the proposal of Zone3. Whit this decision, the MAS fulfill its goal of maximize the number of loads served with highest priority.

If there is a tie in the cost, break the tie by picking the proposal that maximizes the priority function. For example, two proposals with the same cost, one need to remove a load with priority 2 and the other one a load with priority 3. The second proposal is selected because maximizes the priority function.

If a tie persists, select the proposal of the connection which comes first in the index order. For example, two proposals with the same cost, one using the connection of Port Bus (index = 0) and the other one using the connection of Starboard Bus (index = 1). The second proposal is selected because the bus has the lower index.

If a tie persists, select the proposal that arrived first.

3.6.4 Principal Algorithms for MAS Negotiation

As mentioned in the Chapter 2, the actual tasks an agent performs are typically "behaviors". In our case, each ZoneAgent has two principal behaviors for Negotiation purposes, "ContractNetInitiator" and "ContractNetResponder"; these behaviors are based on the interaction standardized by the Contract Net Interaction Protocol. Next we will

present flowcharts of these two behaviors. The corresponding pseudo-code will be presented in Appendix B.

Contract Net Initiator. The flowchart of the Contract Net Initiator is presented in Figures 3.9 and 3.10.

Contract Net Responder. The flowchart of the Contract Net Responder is presented in Figures 3.11 and 3.12.

3.7 Example of the MultiAgent System operation

We use a simple Single Bus Zonal architecture electrical system to show the use of the MultiAgent system. This electrical model has some restrictions:

- Each Ship Service Converter Module(SSCM) has enough capacity to feed one and a half loads.
- If one SSCM is damaged, the remaining two zones can feed the complete system (three loads).
- Each load has a priority. The highest priority is for Zone1 and the lowest is for Zone3.

The initial state of the electrical model is presented in Figure 3.13, where the green lines show the energized lines and the red ones show the not energized lines. We present the operation of the system after three continuous fault scenarios: first a fault in the SSCM of Zone3, then a fault in the SSCM of Zone2 and finally, the fault in the SSCM of Zone3 is cleared.

Fault in the SSCM of Zone3.

In this case, we are using a single bus system, so the zone does not have other connection available and according to the process explained in Section 3.5.2, the ZoneAgent

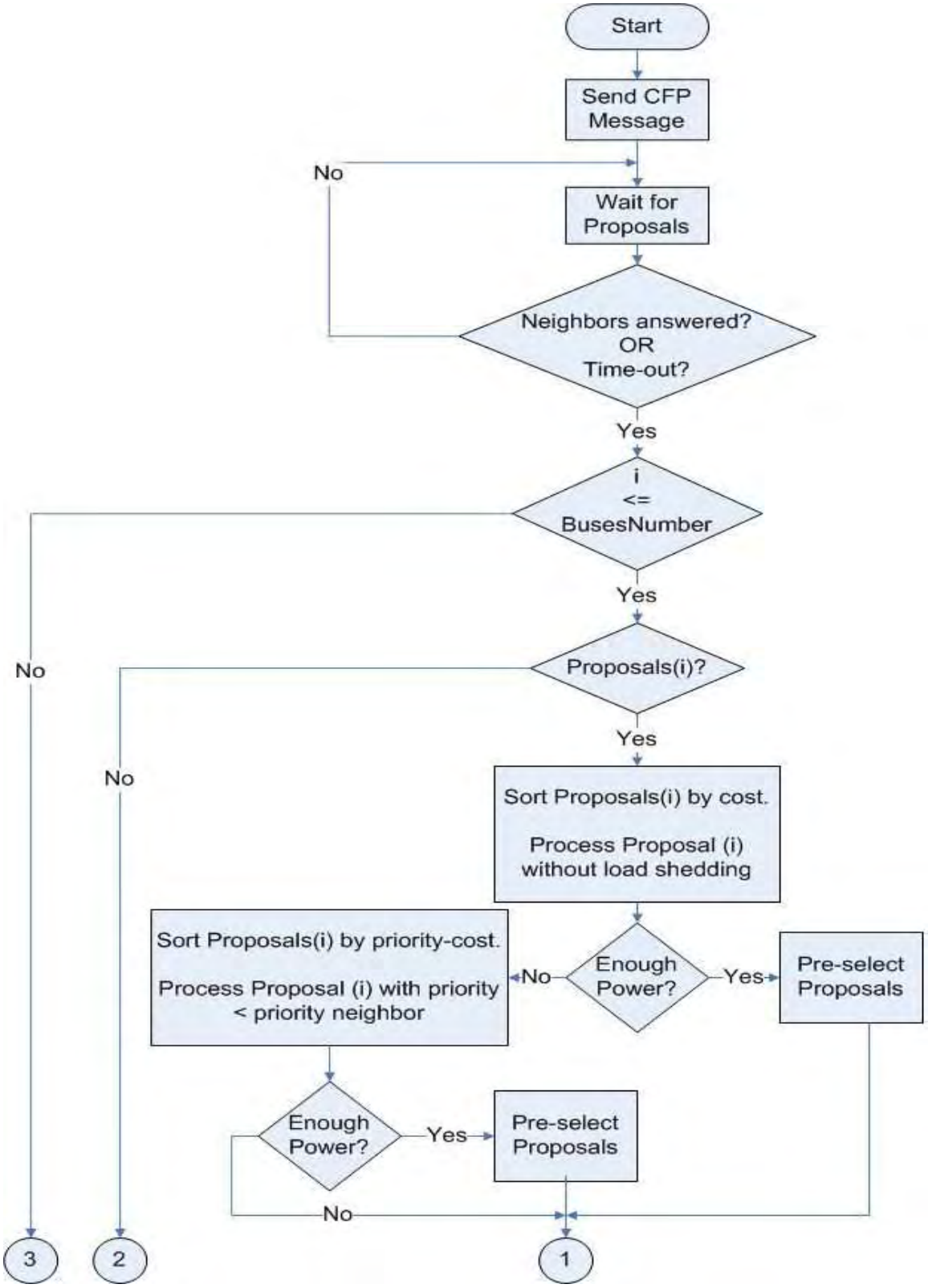


Figure 3.9: Flowchart for Contract Net Initiator (Part 1).

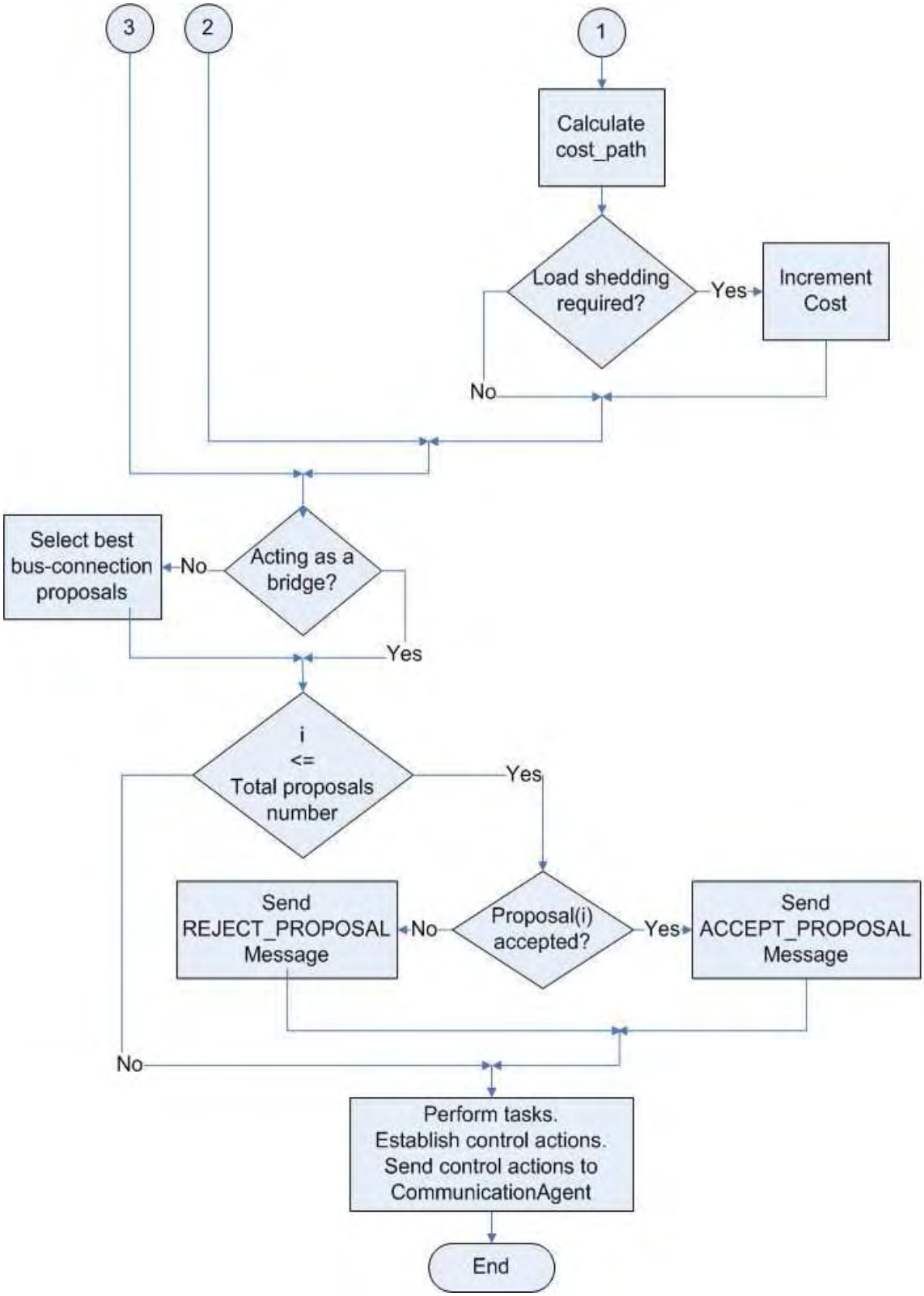


Figure 3.10: Flowchart for Contract Net Initiator (Part 2).

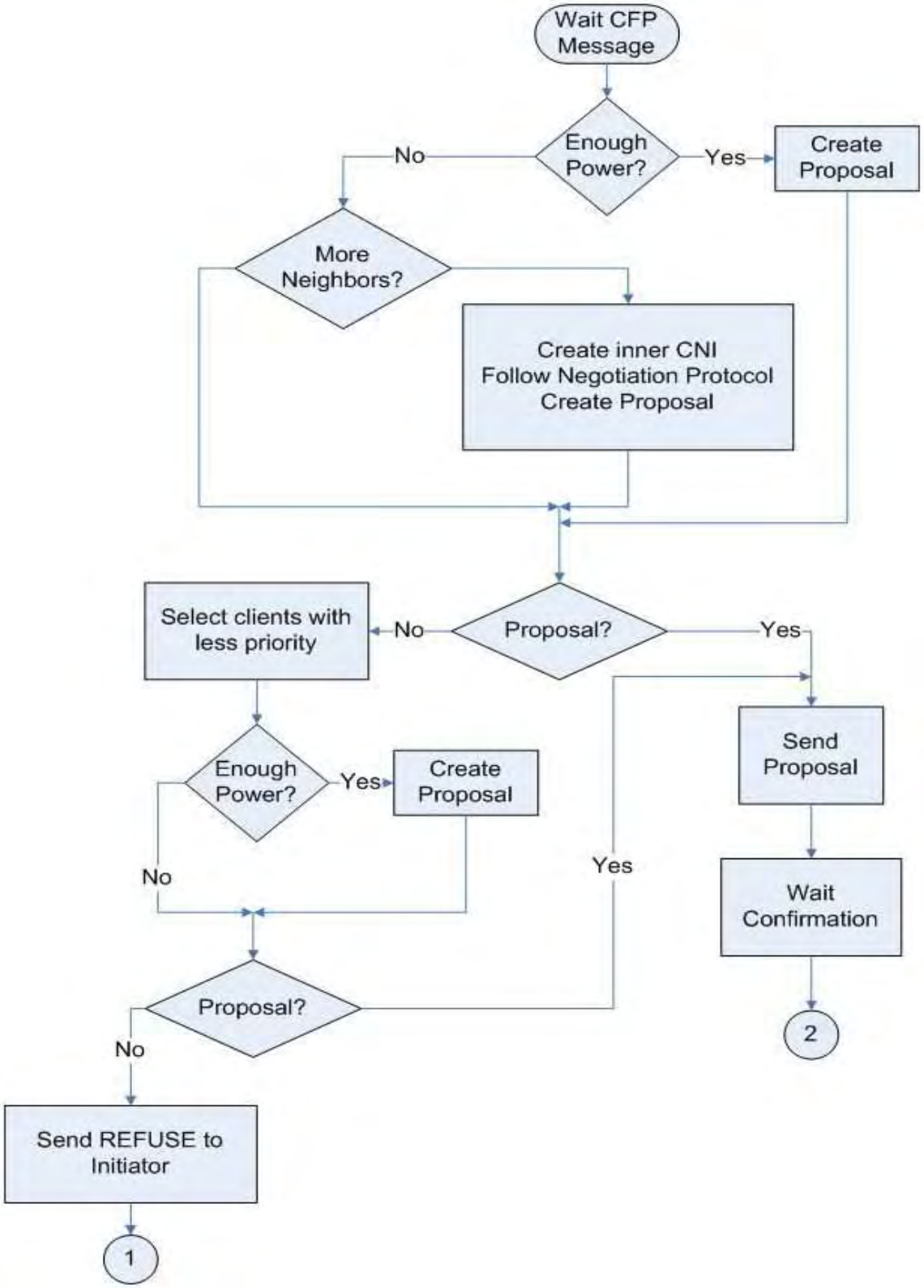


Figure 3.11: Flowchart for Contract Net Responder (Part 1).

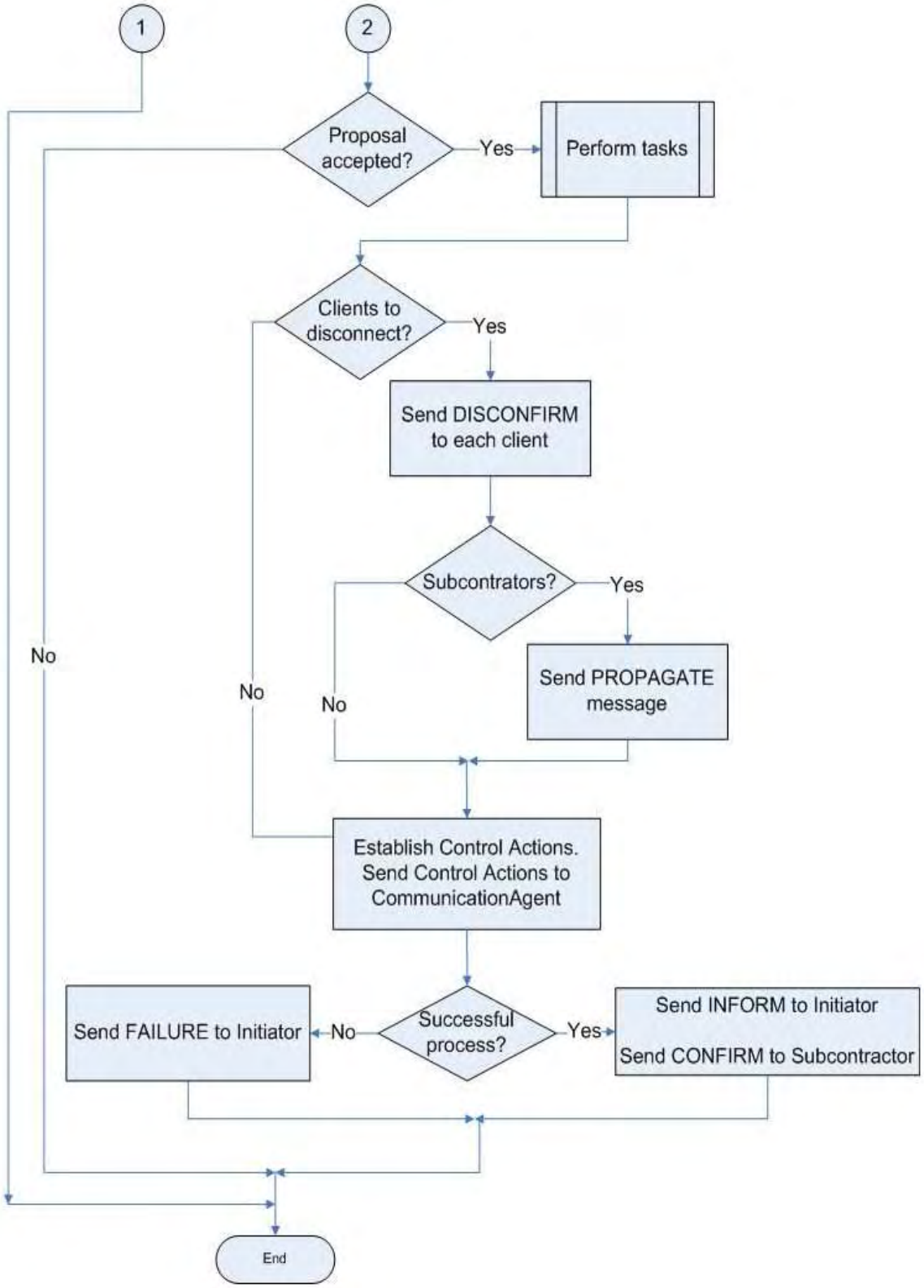


Figure 3.12: Flowchart for Contract Net Responder (Part 2).

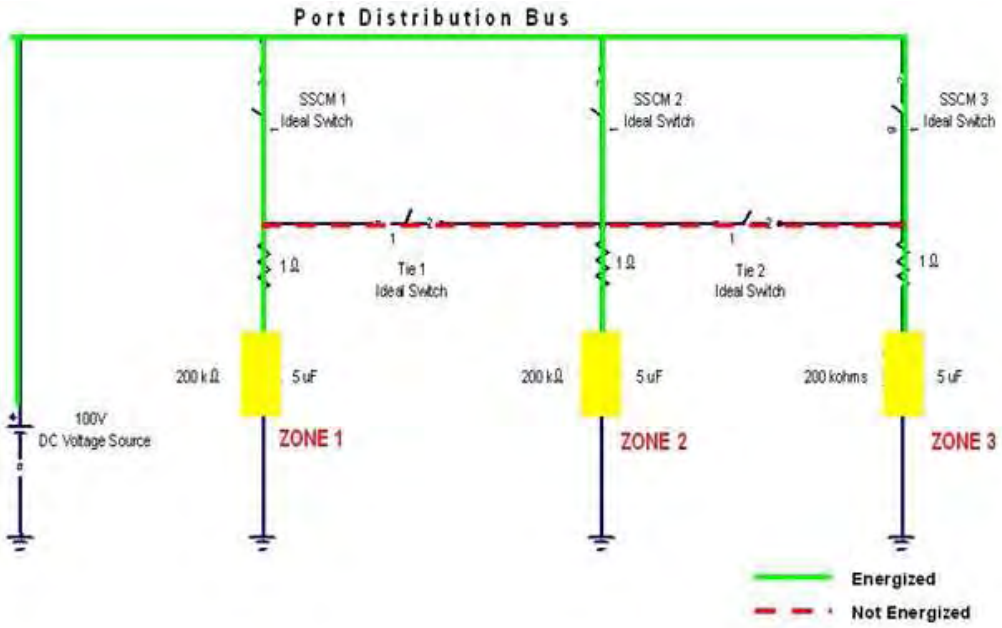


Figure 3.13: MAS Operation Example (Initial State).

must initiate a Negotiation Process.

Figure 3.14 presents the negotiation process to reconfigure the system after the fault in SSCM of Zone3, which is performed as follows:

1. The ZoneAgent3 sends a call for proposals (CFP) message to ZoneAgent2.
2. ZoneAgent2 does not have enough resources to feed both zones, so it sends a CFP message to ZoneAgent1. ZoneAgent2 only asks for part of the power that its zone cannot supply.
3. Zone1 has the capacity to supply the resources that ZoneAgent2 requests. According to Contract Net Protocol, ZoneAgent1 sends a proposal to ZoneAgent2 using a PROPOSE message.
4. With the resources of Zone1 and its own resources, ZoneAgent2 sends a proposal to ZoneAgent3.

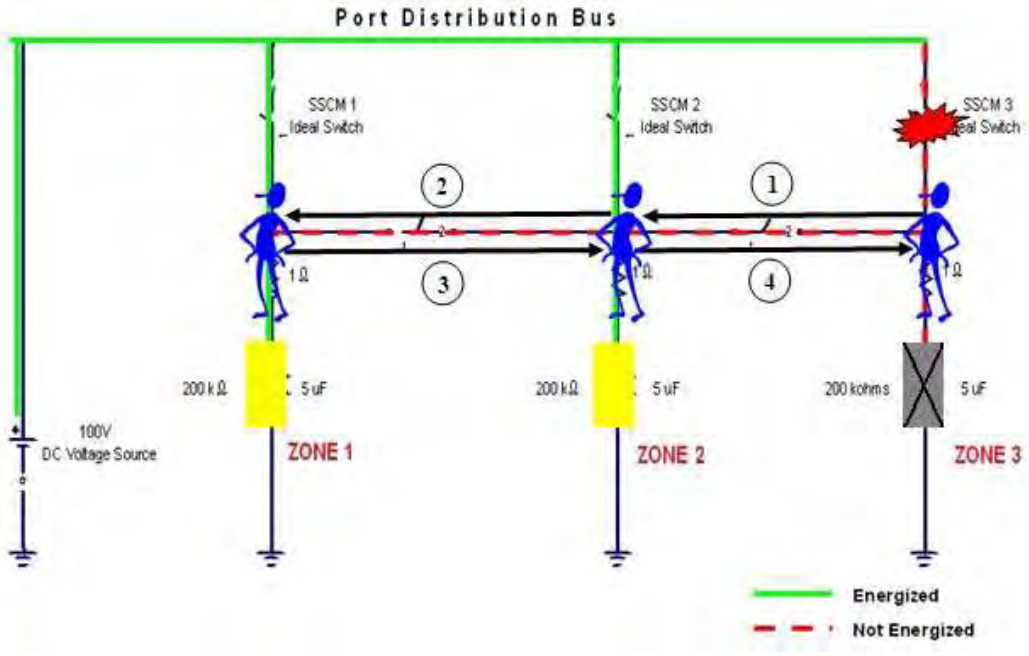


Figure 3.14: MAS Operation Example (Fault in the SSCM of Zone3).

ZoneAgent3 accepts the proposal, and the rest of the process occurs according to Contract Net Protocol. ZoneAgent3 is the client of ZoneAgent2. ZoneAgent2 is a helper to ZoneAgent3, but as a part of this task, it is also a bridge to ZoneAgent1, because part of the resources are from Zone1.

Zone3's load is fed by the connection of the other two zones, for which it is necessary to close the switches between zones. Each ZoneAgent sends the new control actions using a REQUEST message to its CommunicationAgent. Figure 3.15 shows the status of the electrical model after intervention by the MAS.

Fault in the SSCM of Zone2.

From the previous process, ZoneAgent2 maintains ZoneAgent3 as a client. According to the process explained in Section 3.5.2 the same actions must be perform for each client. In this case, the clients of Zone2 are: Zone2 and Zone3.

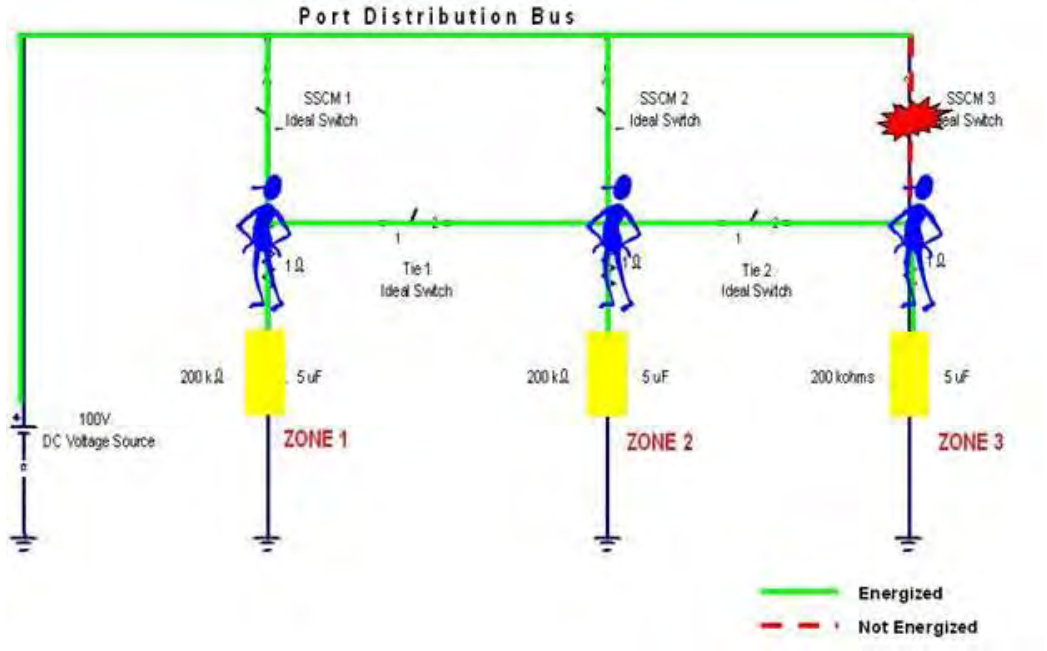


Figure 3.15: MAS Operation Example (Negotiation Results after fault 1).

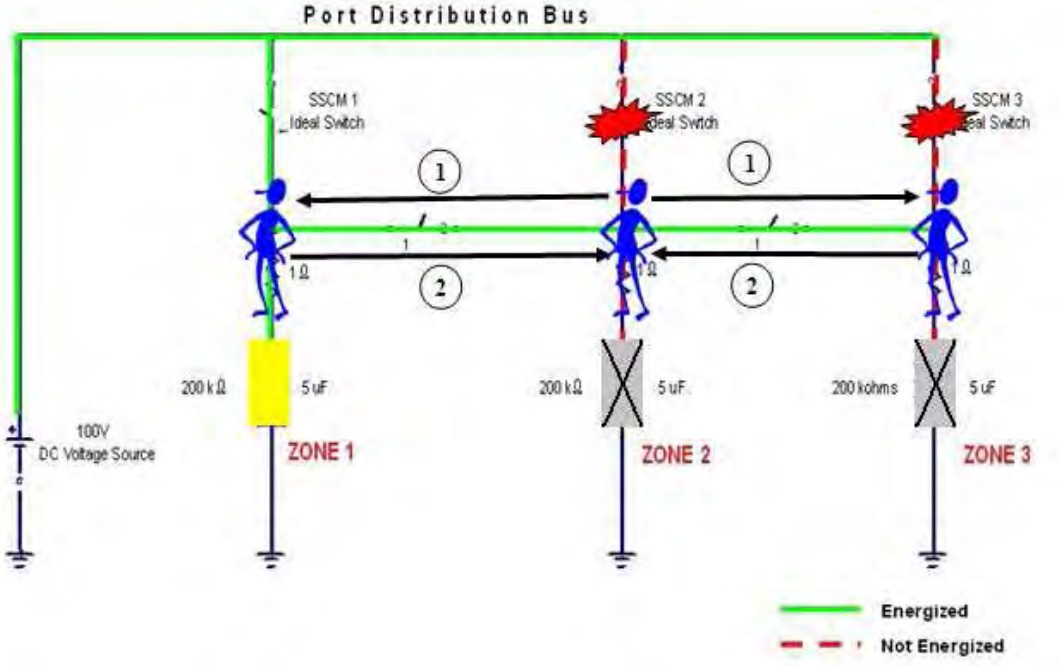
- Client Zone2. Figure 3.16**a** presents the process initiated by ZoneAgent2 to find resources to feed its own load. The process is performed as follows:

  1. ZoneAgent2 sends CFP messages to ZoneAgen1 and ZoneAgent3.
  2. ZoneAgent1 and ZoneAgent3 do not have enough resources and both send a REFUSE message.

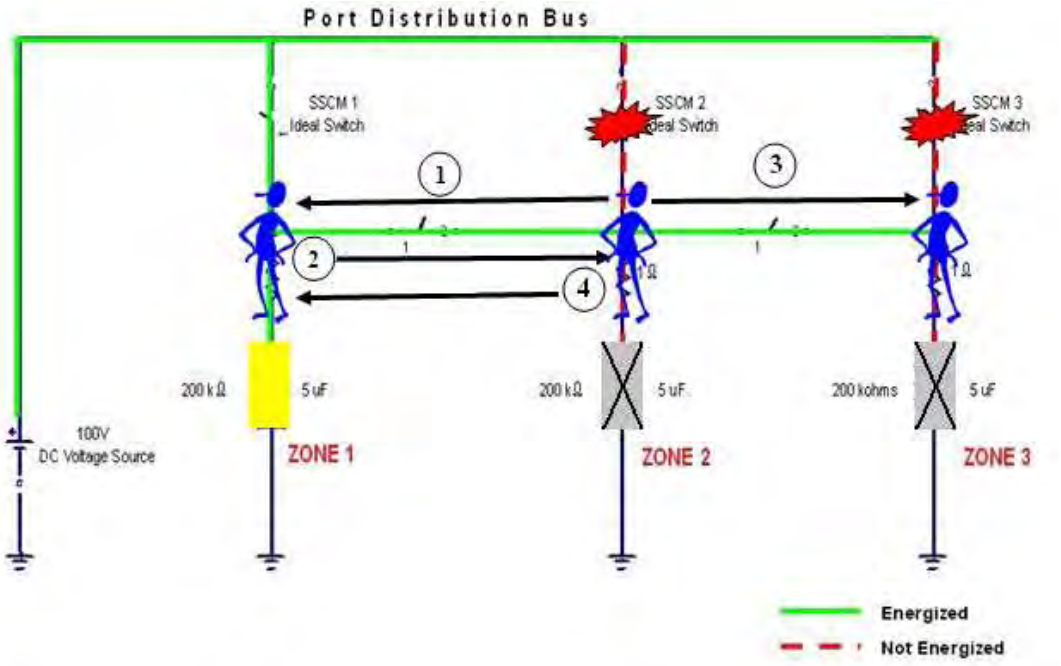
  The negotiation process was not successful, ZoneAgent2 must shed its own load. The zone does not have helpers associated with this client.

- Client Zone3. Figure 3.16**b** presents the process initiated by ZoneAgent2 to find resources to feed the load of its client ZoneAgent3. The process is performed as follows:

  1. ZoneAgent2 sends CFP messages to ZoneAgent1. The CFP is not sent to ZoneAgent3 because it is the client.
  2. ZoneAgent1 does not have enough resources and it sends a REFUSE message.



(a) Process for Client Zone2



(b) Process for Client Zone3

Figure 3.16: MAS Operation Example (Fault in the SSCM of Zone2).

3. The negotiation process was unsuccessful, so ZoneAgent2 sends a DISCONFIRM message to cancel the previous contract that it has with ZoneAgent3.
4. ZoneAgent2 sends a CANCEL message to ZoneAgent1 for releasing resources. ZoneAgent1 was a helper in the proposal with ZoneAgent3. ZoneAgent2 was acting as a bridge.

At this moment, the electrical system can feed only one zone. The highest priority Zone1 is fed from the SSCM of its own zone. The switches between the zones are opened. Figure 3.17 shows the status of the electrical model after the intervention of the MAS.

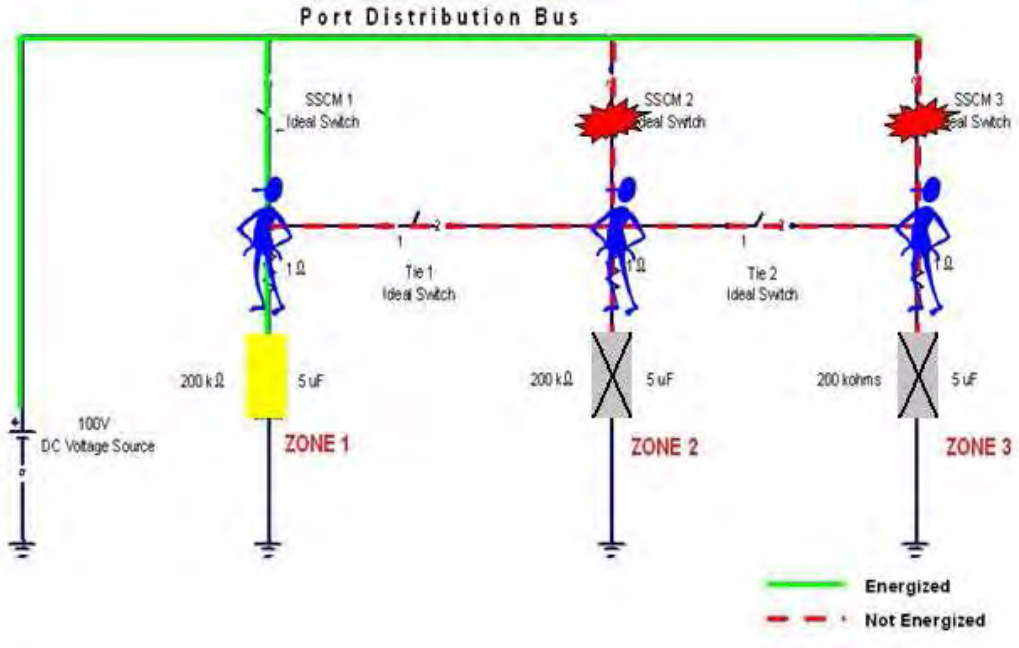


Figure 3.17: MAS Operation Example (Negotiation Results after fault 2).

Fault in the SSCM of Zone3 is cleared.

When the fault in the SSCM of its zone is cleared, ZoneAgent3 connects its load. Then it informs its neighbors and the process to reconfigure the system starts according with the one explained in Section 3.5.3. The process is presented in Figure 3.18 and it is

performed as follows:

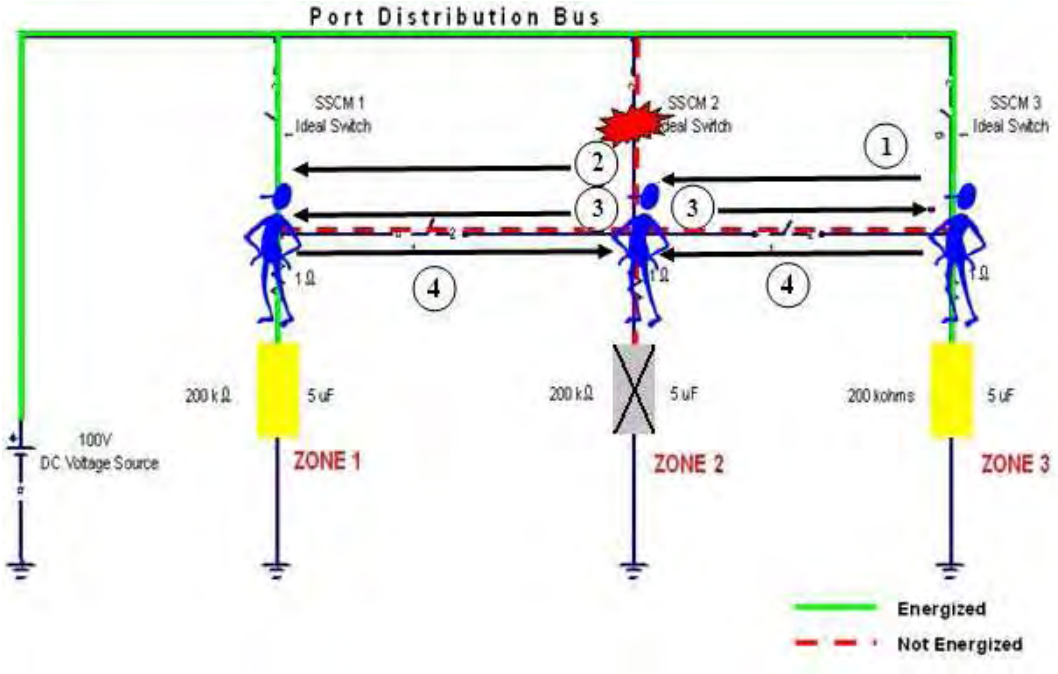


Figure 3.18: MAS Operation Example (Fault in the SSCM of Zone3 is cleared).

1. ZoneAgent3 sends and INFORM_IF message to ZoneAgent2.
2. ZoneAgent2 spreads the information and it sends an INFORM_IF message to ZoneAgen1.
3. ZoneAgent2 needs resources so it sends a CFP to ZoneAgent1 and ZoneAgent3.
4. ZoneAgent1 and ZoneAgent3 send proposals using PROPOSE messages.

Once the negotiation process has finished according to Contract Net protocol, the load of Zone2 is fed by closing the tie switches between the zones. Figure 3.19 shows the status of the electrical model after the operation of the MAS. The system can feed the three loads again.

3.8 Conclusions

Using the concept of IPR, this Chapter presented a distribution of IPRs over a Zonal electrical System. Each zonal IPR is composed by two agents: ZoneAgent and CommunicationAgent. The agents of all Zonal IPRs interact in a MultiAgent System to reconfigure a zonal electric distribution system.

We presented the design of the MAS, which includes the principal messages, interactions, and conversations between agents. It also includes the principal processes that the MAS must perform. At the end of the Chapter, an example of the MAS operation was presented that showed the advantages of using a MAS approach to implement a self-reconfigurable EPDS.

Based on the specifications provides in this Chapter, a prototype of a MAS has been developed and will presented in the next chapter, as an example of how useful a MAS can be in the reconfiguration process.

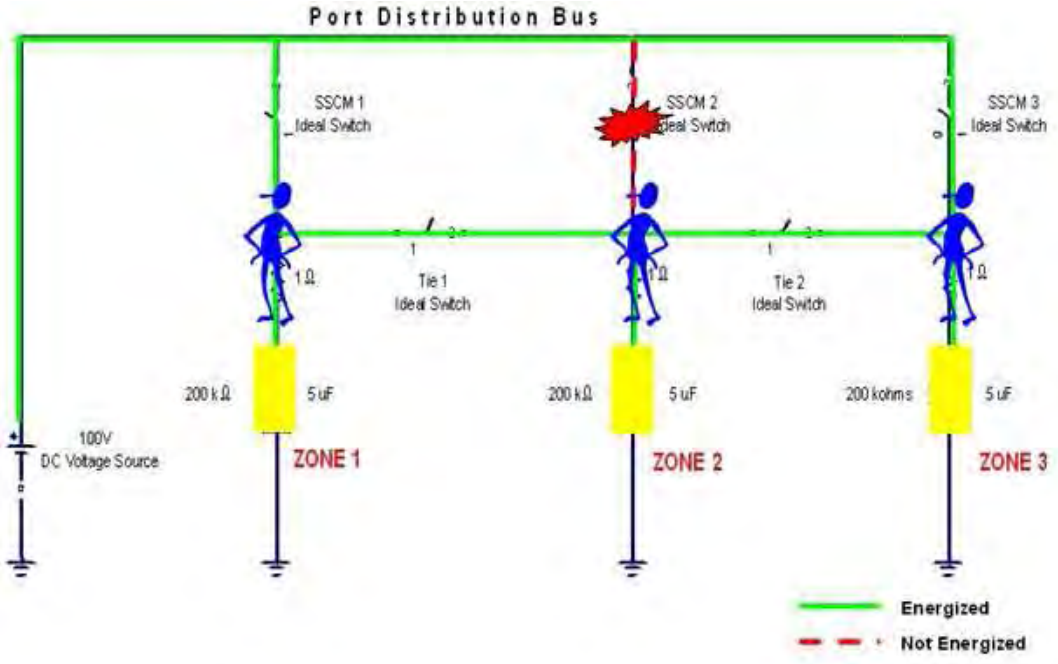


Figure 3.19: MAS Operation Example (Final Results).

CHAPTER 4

Simulation Model

4.1 Overview

This chapter describes the simulation model developed in this work. The implemented simulation model is shown in Figure 4.1. It is composed of three components:

- The Simulink^{TM} [43] implementation of the electric power distribution system (EPDS). We used three electrical models to integrate the MAS, Basic Single Bus DC Zonal System Model, Basic Dual Bus DC Zonal System Model, and Realistic Dual Bus DC Zonal System Model.
- An implementation of the EPDS MAS in the Java^{TM} Agent DEvelopment Framework (JADE) [7].
- Another agent implemented in JADE to handle communication and synchronization between the Matlab ^{TM} model and the EPDS MAS.

The next sections will provide details about the implementation of each component of the Simulation Model.

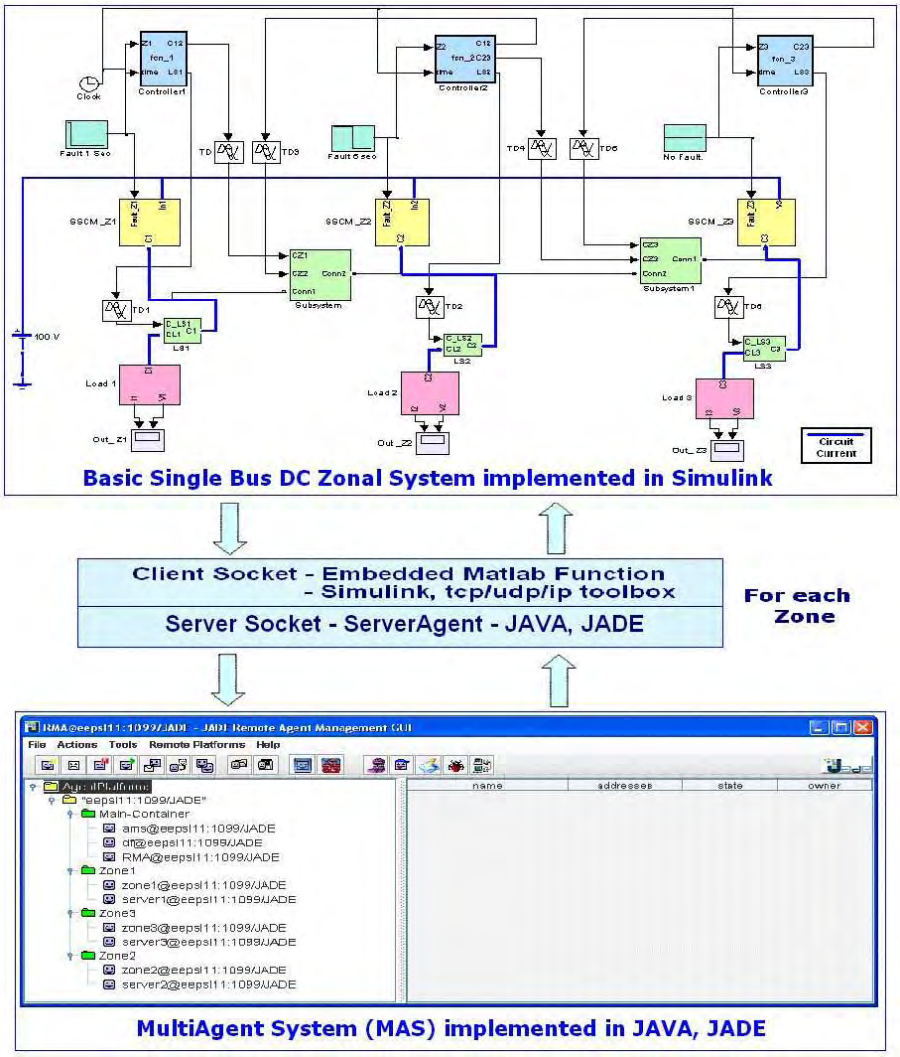


Figure 4.1: Simulation model used in the initial studies.

4.2 Electric Power Distribution System Simulation Model

The implementation of a real EPDS in order to evaluate a MAS is not cost-effective. An alternative is to use a simulation model of the EPDS, in our case we selected Simulink™ because it is a software package for modeling, simulating, and analyzing dynamic systems that supports linear and nonlinear systems, modeled in either continuous time, sampled time, or both. Simulink™ explores more realistic nonlinear models, including specifica-

tions that describe real-world phenomena [44]. Within the Simulink^{TM} environment, we use the SimPowerSystems toolbox to model and simulate the electrical power systems [45]. SimPowerSystems is suitable for developing complex and/or self-contained power systems because it can model the generation, transmission, distribution, and consumption of electrical power, and its conversion into mechanical power [45].

As previously mentioned, due to the advantages of zonal architectures, we selected a zonal distribution to demonstrate the MAS-based EPDS. In Section 2.4.3, we presented the DC-Zonal Electrical Distribution System (DCZEDS), which is based in a Dual Bus DC Zonal Architecture.

We used three test systems based on the DCZEDS, implemented in Simulink^{TM} - SimPowerSystems toolbox. The detailed explanation of each test system is presented next.

4.2.1 Basic Single Bus DC Zonal System Model

For our initial studies, we used a simple circuit of a single bus feeding three loads to emulate the upper half of the DCZEDS. The Basic Single Bus DC Zonal System Model (BSB-DCZS) was developed by Alma Estremera [23] and modified to be integrated with the MAS in this study.

Figure 4.2 shows the BSB-DCZS schematic. The BSB-DCZS [23] is composed of three zones. All loads are simple identical RC circuits. A time constant of $\tau = 1s$ was selected to design the loads. A resistor of $200k\Omega$ was selected to avoid over voltage. A capacitor of $5\mu F$ was obtained from the circuit time equation, $\tau = R \times C$. The power source is a $100V$ DC source. The ship service converter modules (SSCM) are represented by switches that connect the loads to the main bus and tie switches are used to provide further reconfigurability capability to the system.

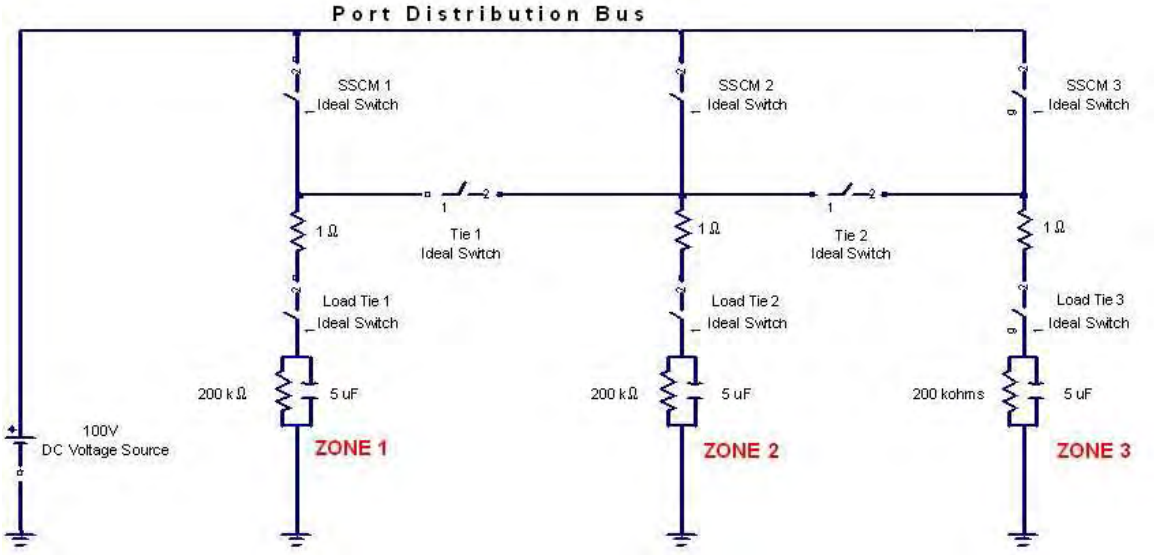


Figure 4.2: Basic Single Bus DC Zonal System Model Schematic.

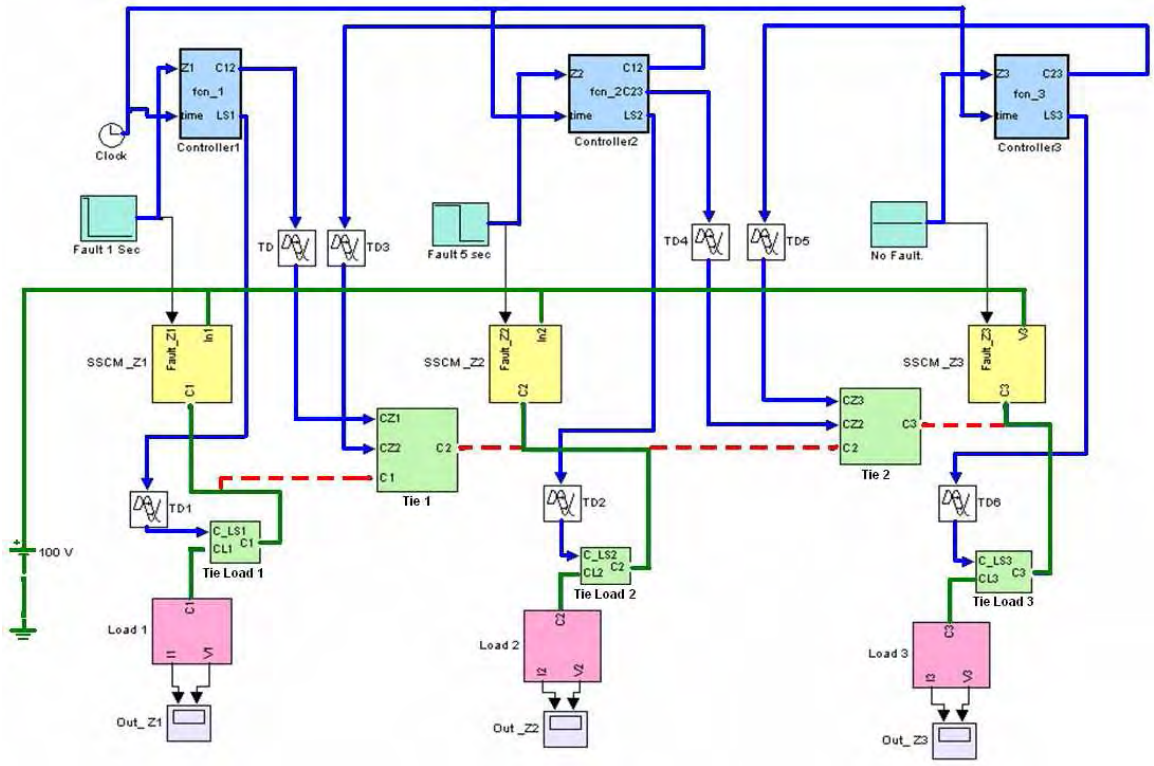
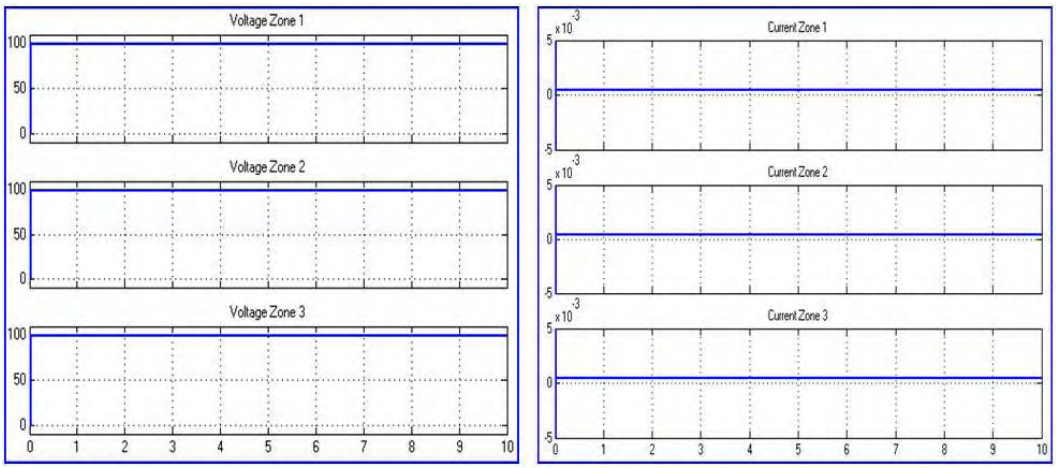


Figure 4.3: Simulink^{TM} Implementation of Basic Single Bus DC Zonal System Model.

Figure 4.3 shows the corresponding Simulink^{TM} - SimPowerSystems toolbox implementation. The DC source energizes the three zones through the bus and is represented by the green lines in this Figure (normal state operation). The blue lines represent the input information that receives the MAS and the output control signals; and the dashed red lines represent the non-energized lines of the system. Figure 4.4 shows sample simulations of the voltage and currents during the normal operation of the system.



(a) Voltage (a) Current

Figure 4.4: Voltages and Currents of BSB-DCZS in Normal Operation.

We have imposed the constraint that the SSCM switches can only support $1\frac{1}{2}$ loads so at least two active SSCM are needed to feed all three loads to model limited power processing capacity for the power converters.

4.2.2 Basic Dual Bus DC Zonal System Model

The second test system used during this research is the Basic Dual Bus DC Zonal System Model (BDB-DCZS). The model was originally developed by Alma Estremera [23] and adapted to be integrated with the MAS for this research.

The BDB-DCZS schematic is shown in Figure 4.5; this model is also a simple

circuit but it is based on a dual bus architecture. The BDB-DCZS is composed of two DC sources to supply three zones. Each zone has two SSCMs, each one is connected to the DC source through a distribution bus. The zones have identical loads with the same parameters that the zones in the BSB-DCZS.

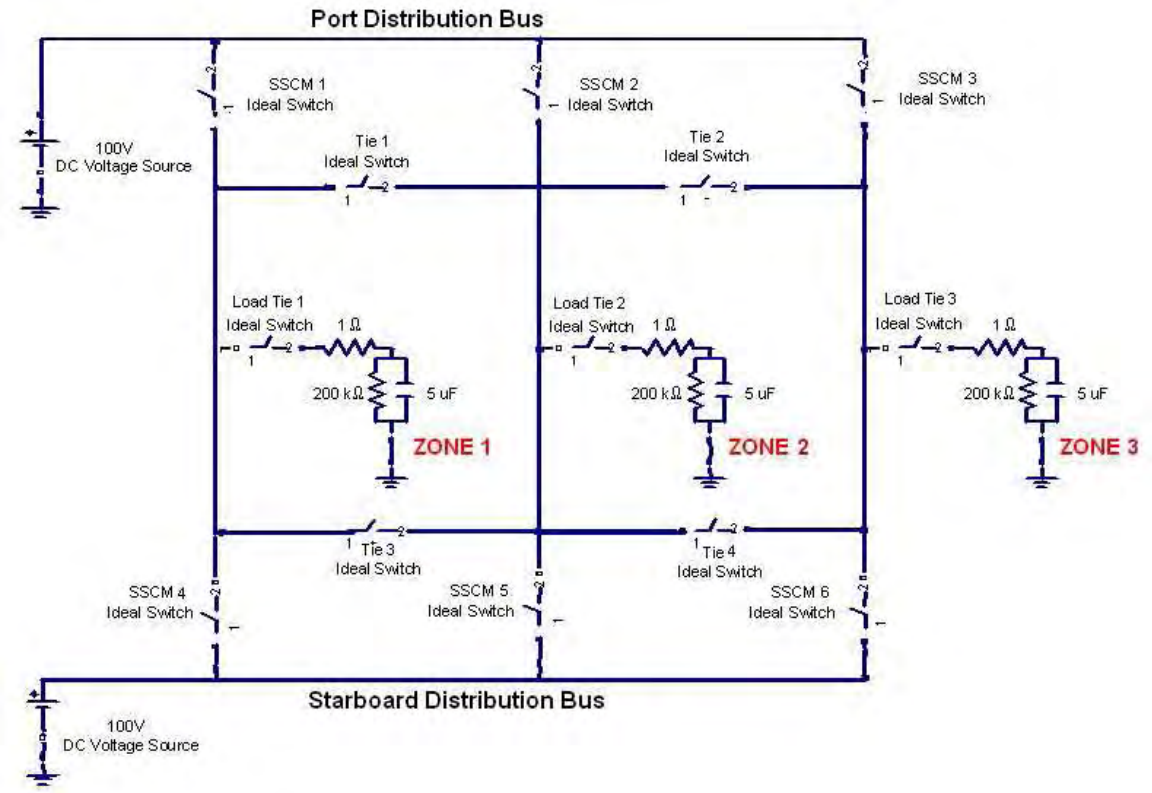


Figure 4.5: Basic Dual Bus DC Zonal System Model Schematic.

Figure 4.6 shows the Simulink^{TM} - SimPowerSystems toolbox implementation of the BDB-DCZS. The green lines show the energized lines of the system during normal operation. The dashed red lines are not energized. As you can see, the lines between the SSCM 3, 4 and 5 (Starboard Distribution Bus) and their loads are not energized because the system is being fed by the SSCM connected to the Port Distribution Bus. Also, the tie switches between the zones are not energized. The blue lines represent the information received by MAS and the output control signals. To simplify the visualization of the system,

some blue lines were replaced by SimPowerSystems toolbox GoTo/From blocks.

The output of voltage and currents of the BDB-DCZS, during normal operation of the system, are basically identical to the output presented in Figure 4.4 of the BSB-DCZS.

When a fault occurs in one of the SSCM connected to the Port Distribution Bus, the first option is to activate the SSCM of the zone connected to the Starboard Distribution Bus. The tie switches between the zones are used for reconfiguration only when a zone has both SSCM in fault condition. This model has the following constraints:

- The SSCM switches can only support $1\frac{1}{2}$ loads so you need at least two active SSCM **connected to the same bus** to feed all three loads to model limited power processing capacity for the power converters.
- A load only can be fed by one bus. For instance, if zone1 and zone3 have faults in their SSCM, the system only counts with the SSCM of zone2. Then, only 2 loads can be fed, one with the SSCM connected to Port Bus and the other one with the SSCM connected to Starboard Bus. In this case, loads of zone1 and zone2 will be fed because they are the ones with the highest priorities.

4.2.3 Realistic Dual Bus DC Zonal System Model

The third model used in our studies is the Realistic Dual Bus DC Zonal System Model (RDB-DCZS). The model was developed by the undergraduate Electrical Engineering student Luis J. Collazo and it is based on the Simulink^{TM} implementation of the DC Zonal Electric Distribution System (DCZEDS) described on [18, 19, 20], which is a simulation testbed for control algorithms for ship power distribution system provided by the Office of Naval Research (ONR) to researchers in the NSF/ONR Electric Power Networks Efficiency and Security (EPNES) program [20]. The schematic of the DCZEDS was presented in Figure 2.5, Section 2.4.3.

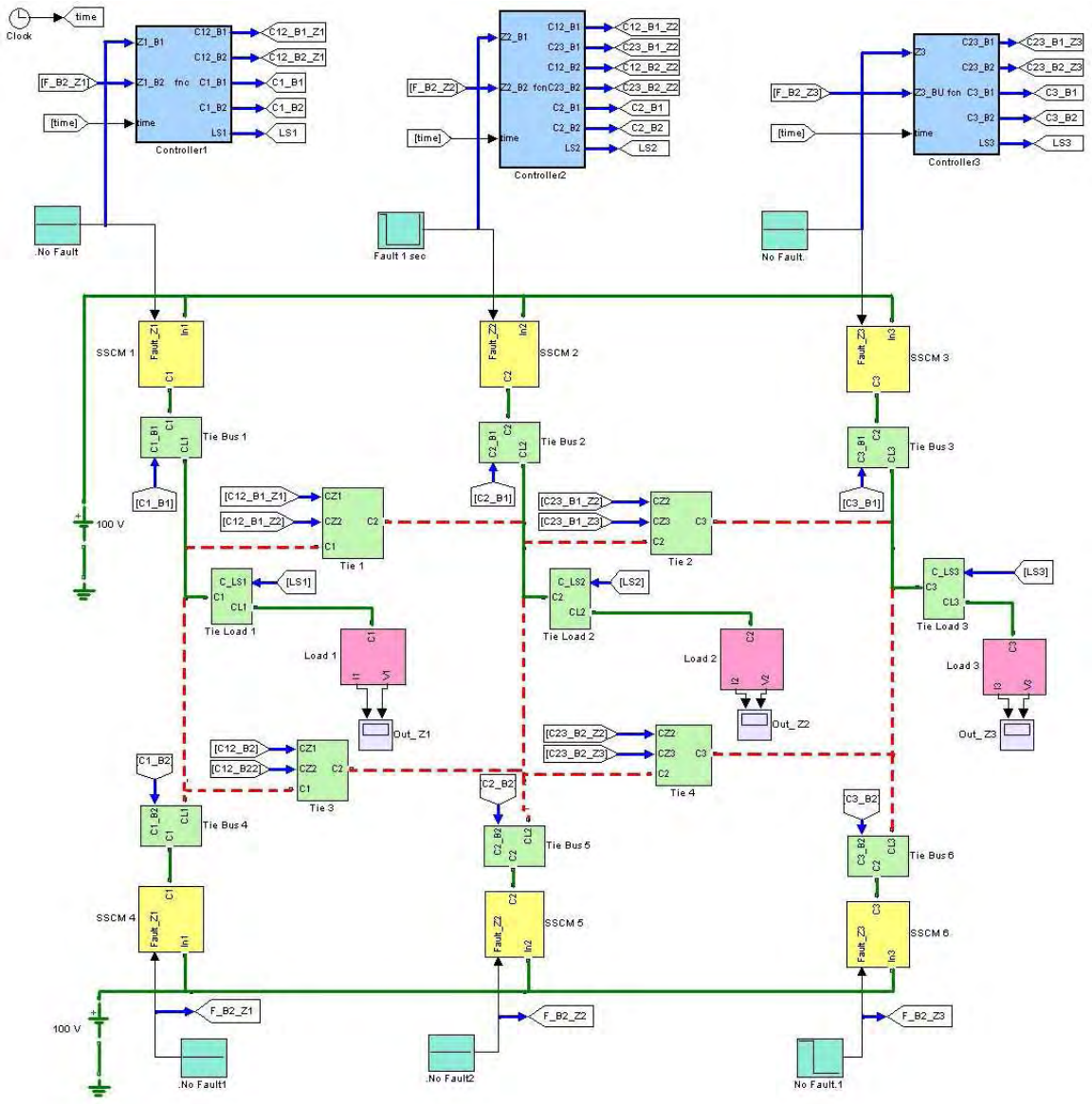


Figure 4.6: Simulink^{TM} Implementation of Basic Dual Bus DC Zonal System Model.

Figure 4.7 shows the Simulink™ - SimPowerSystems toolbox implementation of the RDB-DCZS. The green lines show the energized lines of the system during normal operation, the dashed red lines show the ones. The blue lines represent the input information that receives the MAS and the output control signals that it sends. For simplicity, some blue lines were replaced by SimPowerSystems toolbox GoTo/From blocks. As you can see, the distribution of green, red and blue lines is similar to the one presented in the BDB-DCZS model.

The RDB-DCZS is composed by two power supplies (PS), each one connected to a 500-V bus, one on the Starboard side and one on the Port side. Also, it contains six Ship Service Converter Modules (SSCM) connected to their load using an OR-ing diode. In this model the loads are not identical. The load of Zone 1 is composed by one Ship Service Inverter Module (SSIM) and its associated Load Bank (LB). Zone 2 has a Motor Control (MC) module as load and Zone 3 has a Constant Power Load (CLP). Appendix A presents a description of the loads of each zone and the SSCM.

The output waveforms of the RDB-DCZS, during the normal operation of the system, are presented in Figure 4.8.

This model is constrained by the capacity of each component of the model, and that a load only can be fed by one bus, as it was presented in the constrains of BDB-DCZS.

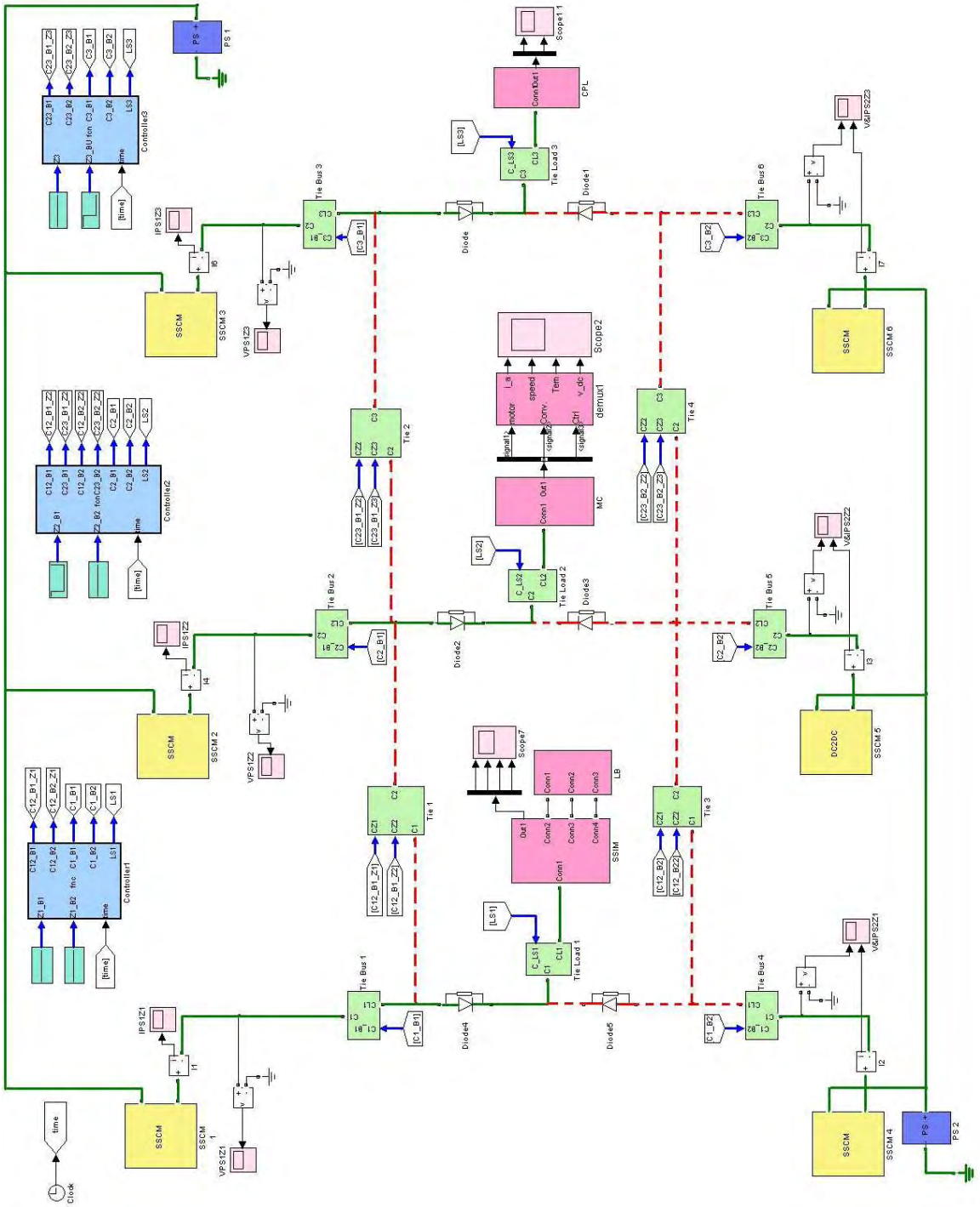


Figure 4.7: Simulink^{TM} Implementation of Realistic Dual Bus DC Zonal System Model.

4.3 MultiAgent System Prototype

The MultiAgent System is implemented using the platform Java™ Agent DEvelopment Framework (JADE) [7], which is a Java™ framework for developing FIPA-compliant agent applications.

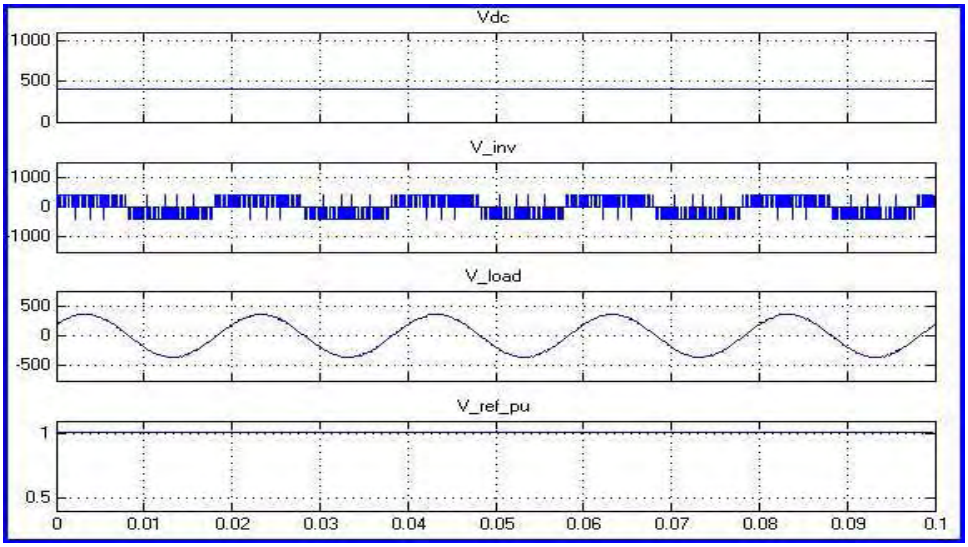
4.3.1 JADE: Java™ Agent DEvelopment Framework

The Java™ Agent DEvelopment Framework (JADE) is a framework that simplifies the development process of FIPA-complaint MAS applications [7, 46] . It is an Open Source project developed by CSELT (Centro Studi e Laboratori Telecomunicacin) of Telecom Italy using the Java™ programming language. It can be downloaded from the JADE Home Page [7].

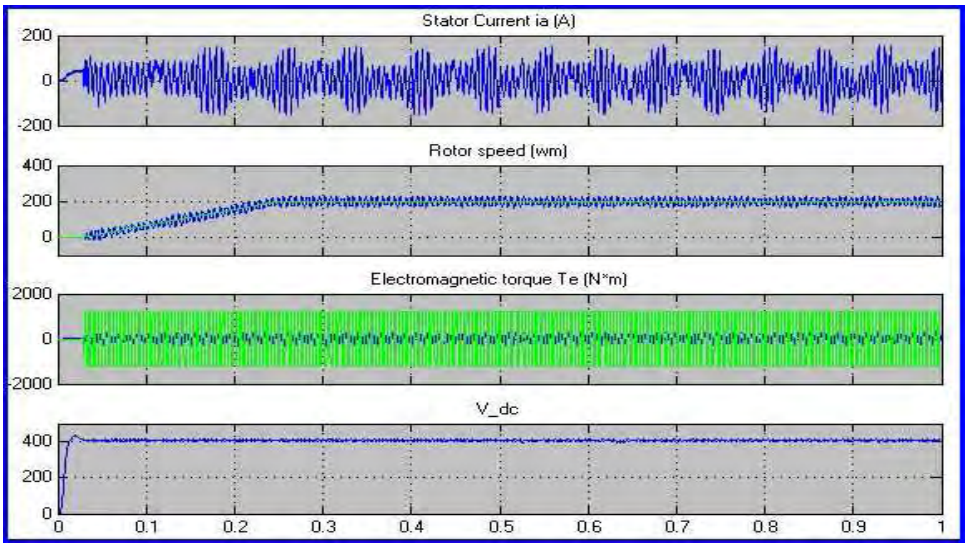
A JADE agent platform is a distributed system that can be split over several hosts. One of the hosts must have the main container where the AMS (Agent Management System) and DF (Directory Facilitator) agents are located. This host acts as a front end and the platform appears as a single entity to the outside world. Each host can have one or more agent containers running over a Java™ Virtual Machine. The architecture of JADE is shown in Figure 4.9.

JADE, following FIPA specifications [6], implements those aspects of a MAS that are not internal particularities of the agent and are independent of the type of application. JADE also provides some tools to facilitate development and debugging processes, and to track the agents' actions. One of these tools is the Sniffer Agent, which tracks and visualizes the message exchange between agents.

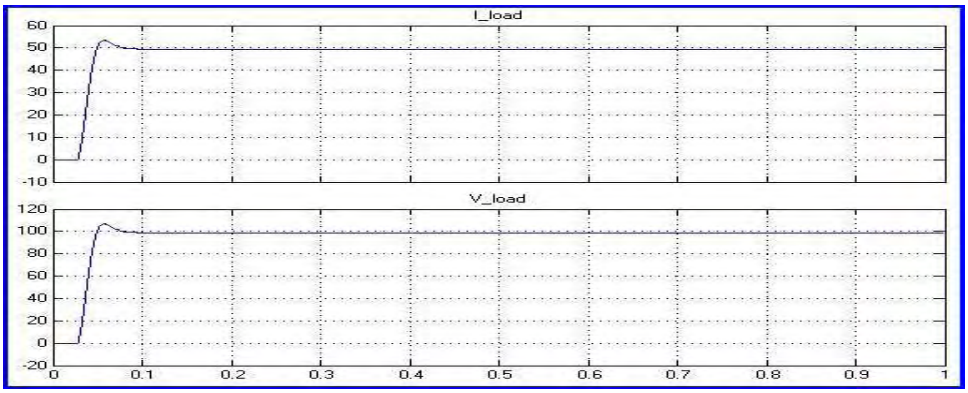
JADE is used mainly because it is a middleware that hides the distributed architecture where the MAS lives. JADE allows the developer to focus on the logical aspect of



(a) Zone 1 in Normal Operation.



(b) Zone 2 in Normal Operation.



(c) Zone 3 in Normal Operation.

Figure 4.8: Output waveforms of RDB-DCZS in Normal Operation.

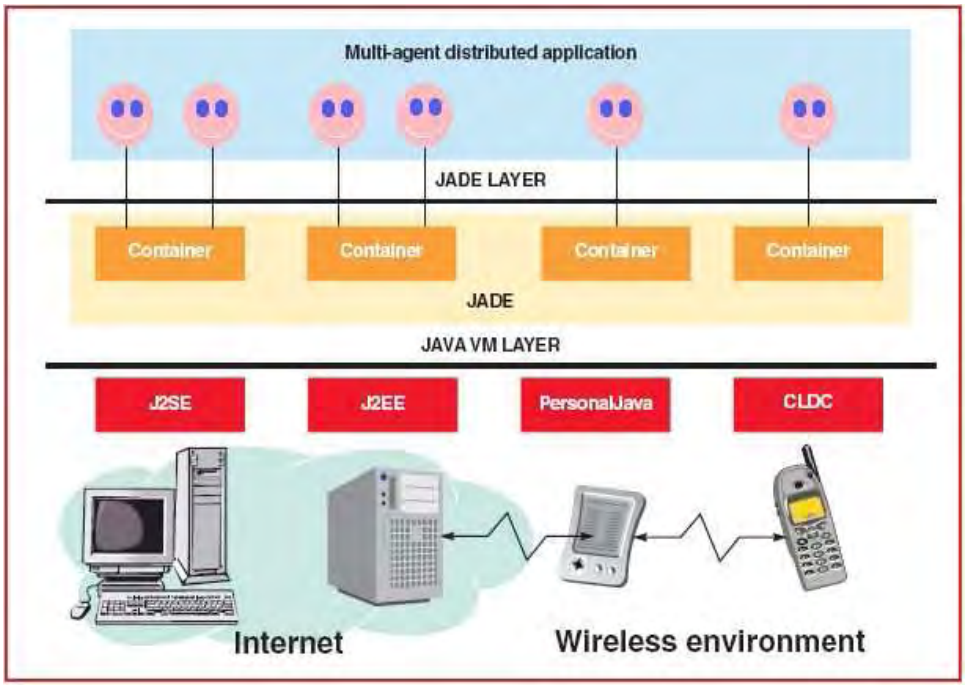


Figure 4.9: JADE architecture [47].

the system because it controls the communications between different hosts and the message exchange between agents. Other reasons that make JADE a suitable platform for developing MAS applications, can be found in [47].

4.3.2 MAS implementation

The MAS prototype (lower part of Figure 4.1) was developed according to the specifications provides in Chapter 3. Figure 3.2 presents the MAS-based proposed Architecture and contains some elements required by FIPA: Agent Management System (AMS), Agent Directory Facilitator (DF) and a Message Transport Service (MTS). These elements will not be implemented because they are provided by JADE. The implementation will focus on the agents that compose the Zonal_IPR. The MAS is composed of two principal agents for each zone, ZoneAgent and ServerAgent.

The ZoneAgent has the needed behaviors for initiating and responding a Contract

Net Protocol. Also, it can listen and answer messages that are not part of the CNP.

The ServerAgent is the implementation of the CommunicationAgent, but its name was changed because it performs a server function in the Communication Middleware. The ServerAgent provides the bridge between ZoneAgent and Simulink™ model. The ServerAgent has behaviors to listen and responde messages. It can also act as a TCP/IP Server, which will be explained in next section.

During simulation time, a communication channel is established between the Simulink™ model and ServerAgent. When the ServerAgent detects a change in system conditions, it sends this information to the ZoneAgent. In this version of the prototype, only faults on the SSCMs are considered.

The ZoneAgent negotiates with its neighbors to reconfigure the system. Each ZoneAgent negotiates only with its immediate neighbors, reducing the dependency between the algorithm and the system topology. The interaction between agents is based on the specifications provided by the Foundation for Intelligent Physical Agents (FIPA) such as Agent Communication Language (ACL) [13] and Contract Net interaction protocol [42, 41].

4.4 Communication Middleware

In this section, we present the detailed implementation of the communication middleware used in the Simulation Model. The communication middleware allows the MAS to send/receive data to/from the EPDS model in Simulink™. The MAS needs to know the status of the SSCM in the zonal power system to take the necessary control actions, that are applied to the electric model.

4.4.1 Simulation package: Matlab™ Simulink™

Simulink™ - SimPowerSystems toolbox was used to simulate the EDPS. The top part of Figure 4.1 shows the Simulink™ - SimPowerSystems implementation of the BSB-DCZS (This part changes according to the test system used in each case). The model includes one "Embedded Matlab™ Function" block for each Zone, these blocks send/receive information to/from the MAS.

4.4.2 Embedded Matlab™ Functions Limitations

As mentioned in the previous Chapter the MAS is implemented using the JADE platform and Java™ programming language in order to make it FIPA-complaint. It was also mentioned that Simulink™ is integrated with Matlab™. Matlab™ supports Java™ [48].

In our initials implementation we tried to use Java™ functions to communicate with JADE in the "Embedded Matlab™ Function" unsuccessfully. This occurred because one of the unsupported features of the "Embedded Matlab™ Function" is Java™, although Matlab™ supports Java™ [44]. The alternative was to implement the middleware using TCP sockets.

4.4.3 Client/Server Socket communications

A socket is one endpoint of a two-way communication link between two programs running on the network and bound to a port number [49].

A socket can perform seven basic operations [50]:

- Connect to a remote machine
- Send data

- Receive data
- Close a connection
- Bind to a port
- Listen for incoming data
- Accept connections from remote machines on the bound port

To use sockets, it is necessary to have a Server and a Client. In our Simulation Model, the Client is the implementation in Simulink™ and the Server is the MultiAgent System.

4.4.3.1 Server Socket.

As a Server Socket, we implement the CommunicationAgent referred to as a "ServerAgent". The ServerAgent is an agent that has the function to communicate with remote clients via a socket. When the ServerAgent is launched, it searches for a file with the initialization parameters, like name and port number to which it listens for connections. Each message that arrives through the socket is analyzed and if the conditions of the system have changed a message with this information is sent to the ZoneAgent. Each control action from the ZoneAgent is forwarded back to the EPDS model through the socket again.

4.4.3.2 Client Socket.

To open a Client Socket in Simulink™, we needed Matlab™ code, so we used an "Embedded Matlab™ function". We used "TCP/UDP/IP Toolbox" developed by Peter Rydester [51]. To set up TCP/IP connections in Matlab™ and to transmit data over a socket between Matlab™ processes and other applications. It is possible to act as server and/or client and transmit textstrings, arrays of any datatype, files or Matlab™ variables.

4.4.3.3 Client/Server Operation.

Figure 4.10 shows the process of requesting a connection. When the MultiAgent System starts, the ServerAgent of each zone is listening to a specific port number and waits for a client to make a connection request (in this case the client is the "Embedded Matlab™ function" of the zone). The command used to create a Server Socket is:

```
ServerSocket listen_socket = new ServerSocket(port);
```

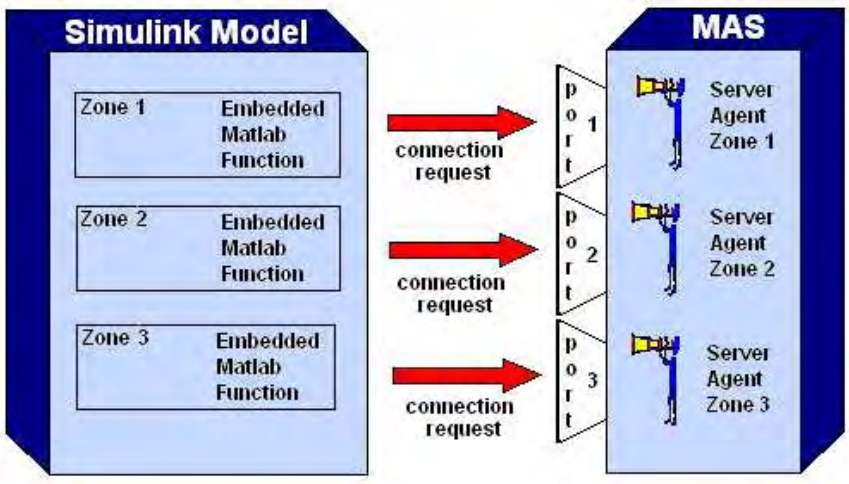


Figure 4.10: Request a connection to the ServerAgent.

When the "Embedded Matlab™ function" in the Simulink™ Model asks for a connection, the ServerAgent must accept it. The commands used in the ServerAgent to accept and to establish input and output streams are:

```
Socket client_socket = listen_socket.accept();

BufferedReader in = new BufferedReader
    (new InputStreamReader(client_socket.getInputStream()));

PrintStream out = new PrintStream
    (client_socket.getOutputStream(),true);
```

On the client-side (Model in Simulink™), the client knows the host name of the machine on which the ServerAgent is running and its port number. At the beginning of the simulation, a connection to the ServerAgent is created using the function "PNET" provided by "TCP/UDP/IP Toolbox" with the following instructions:

```
if (isempty(con))
    con = pnet('tcpconnect', 'localhost', 4096);
end
```

The ServerAgent accepts the connection. On the client side, when the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server. The client and server can now communicate by writing to or reading from their sockets as is shown in Figure 4.11.

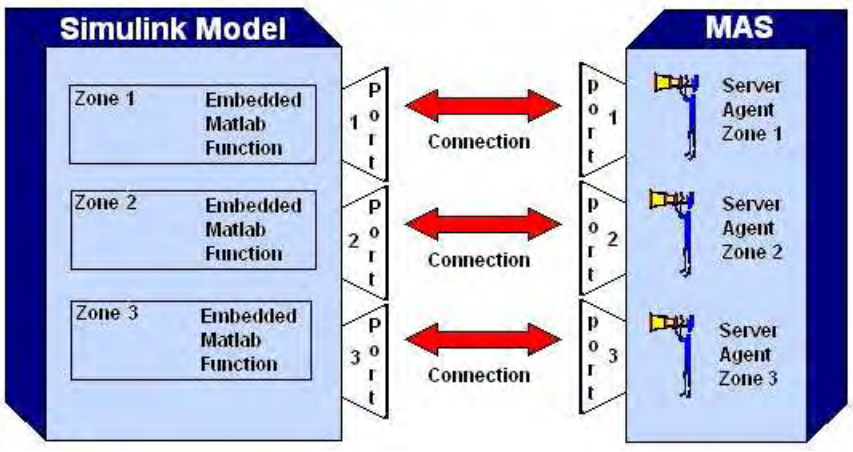


Figure 4.11: Client/Server Connection Established.

The commands used in the "Embedded Matlab™ Functions" to send/receive information to/from the ServerAgents are:

- Send Information:

```
string = num2str(time); pnet(con, 'printf', string, 'noblock');
```

- Receive Information:

```
str = pnet(con, 'readline', 'noblock');
```

```
C12 = str2double(str);
```

In both cases, the parameter "noblock" allows the simulation to continue even if the answer of the MAS has not been sent; this is because in a real operation the EPDS will continue with or without the answer from the controller. The commands used in the ServerAgent to send/receive information to/from the Zone in the Simulink™ Model are:

- Send Information:

```
out.println("control");
out.flush();
```

- Receive Information:

```
double time = Double.parseDouble(in.readLine());
```

4.4.4 Synchronization problems

In our first experiments, we used a "Continuous simulation type" in the Simulink™ Model and the MAS received information of SSCM status, processed them and sent an answer without considering the "simulation time" in which the message was received and sent. Through experiments, we found that Simulink™ changes the step size of the time in runtime, and when the conditions of the simulation change (fault, recovery or control actions) it goes back in simulation time and it uses a small step. For this reason, sometimes the MAS received information in an specified time and when the control actions were applied in the model the time of the simulation was before than the time the message was sent.

Our first approach to solve this synchronization problem was to send the time of the simulation in each message between the Simulink™ Model and the ServerAgent. Before the ServerAgent replied, it verified the time was later than the simulation time of the request.

```
If(currentTime>requestTime) sendAnswer();
```

But this approach had some drawbacks because it could cause inaccurate results. Therefore, besides the control in the simulation time, we decided to use a "Discrete Simulation Type" in the Simulink^{TM} Models. We selected "Discretize electrical model" as a "Simulation type"; this is one of the methods to solve a circuit provided by SimPowerSystems toolbox. "The electrical system is discretized using the Tustin method, which is equivalent to a fixed-step trapezoidal integration. To avoid algebraic loops, the electrical machines are discretized using the Forward Euler method" [45]. This new simulation type helped us synchronize the MAS and the power system simulation because we can avoid the loops in the simulation time explained previously.

4.5 Conclusions

This Chapter presented details about the implementation of the simulation model used in this research. The simulation model is composed of three components: the simulink implementation of each power system used as a test system, the MAS implemented in JADE, and the communication middleware that communicates the other two components.

The prototype of the MAS was developed and integrated with the test systems. Each test system was modified to interac with the MAS and to allow reconfiguration. The simulation model must be tested in order to validate the reconfiguration logic. The next chapter will present the simulation results for each test system.

CHAPTER 5

Simulation Results

5.1 Overview

Based on the specifications of a MAS-based EPDS and the electrical simulation test systems presented in Chapters 3 and 4; a MAS prototype was implemented and integrated with three electric simulation models. It is important to remember that the objective of the MAS is to maximize the number of served loads with highest priority during a fault condition.

In order to test the operation of the MAS and to evaluate its performance, some scenarios were simulated and evaluated. Next, we will present, the results of two of the scenarios for each of the test systems. We include the description of the message exchange of agents using the Sniffer Agent GUI.

The priority scheme is: Zone1 has the highest priority 1, Zone2 has priority 2, and Zone3 the lowest priority 3. First, each scenario is simulated using this priority scheme. Then, the priorities are changed and the scenario is simulated again. This process is made to show independence between the configuration of the EPDS and the MAS operation. Notice that we keep the parameters of each scenario; we only change the priority scheme.

5.2 Simulations results for BSB-DCZS

The first test system is the Basic Single Bus DC Zonal System Model (BSB-DCZS) introduced in Section 4.2.1. The BSB-DCZS is a simple circuit of a single bus feeding three identical RC loads. Each SSCM switch can only support $1\frac{1}{2}$ loads so at least two active SSCM are needed to feed all three loads.

5.2.1 BSB-DCZS - Scenario I

The general parameters for Scenario I are:

- Duration of Simulation: 10 seconds.
- At the beginning of the simulation the system is in normal state (no faults).
- At t=1.5 seconds, a fault takes place in the SSCM switch that feeds Zone1. The MAS must respond to this event and reconfigure the system, so the load of Zone1 is fed by the connection of the other two zones, for this, it is necessary to close the switches between zones.
- At t=3.5 seconds, a fault takes place in the SSCM of Zone2. At this moment, the system can feed only one Zone, so the highest priority Zone is selected.
- At t=5.5 seconds, the fault in the SSCM of Zone1 is cleared. The system can feed the three loads again. The load of Zone2 is fed using the tie switches, so the switches between all zones are closed.
- At t=6.5 seconds, the fault in the SSCM of Zone2 is cleared and the system is restored to its initial no-faults state.

5.2.1.1 Test with Default Priority Scheme

For the first simulation of this scenario, we used the default priority scheme. Figures 5.1 and 5.2 show the Sniffer Agent GUI (Part 1 and Part 2).

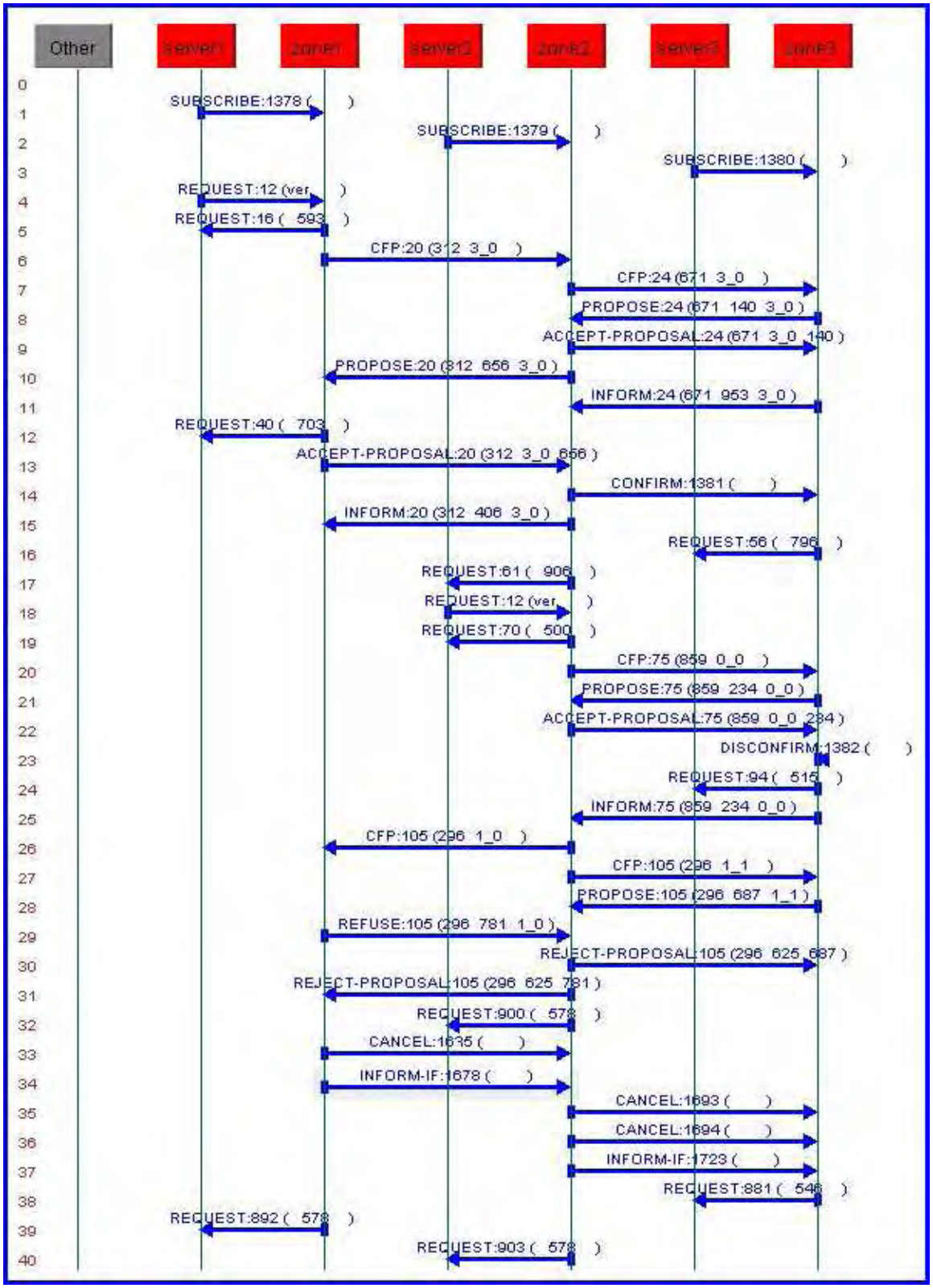


Figure 5.1: BSB-DCZS Scenario I - Default Priority: Sniffer Agent GUI - Negotiation Process (Part 1).

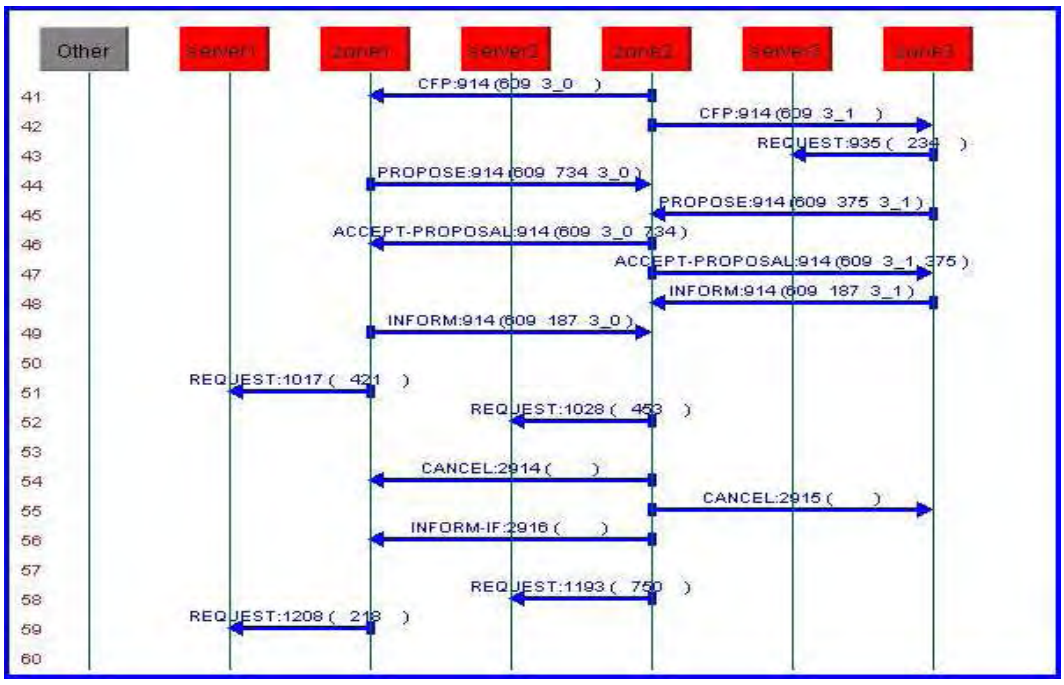


Figure 5.2: BSB-DCZS Scenario I - Default Priority: Sniffer Agent GUI - Negotiation Process (Part 2).

When the system starts, the connections between the zones in the Simulink™ model and their ServerAgents in the MAS are opened. Then, each ServerAgent sends a SUBSCRIBE message to its ZoneAgent. During the time of the simulation the client socket in the Simulink™ Model sends information regarding the status of the SSCM to the ServerAgent.

When a fault in Zone1 occurs at t=1.5 sec., the ServerAgent1 sends a REQUEST message to ZoneAgent1 to inform the agent about the fault. In representation of its zone, ZoneAgent1 sends a call for proposals (CFP) message to its neighbor ZoneAgent2. ZoneAgent2 checks the available resources of its zone, but does not enough to feed both zones. ZoneAgent2 needs to obtain part of the power requested by ZoneAgent1; so it sends a CFP to its neighbor ZoneAgent3. According to Contract Net Protocol, ZoneAgent3 sends a PROPOSE message to ZoneAgent2, and this proposal is accepted by ZoneAgent2 using an ACCEPT-PROPOSAL message. When ZoneAgent2 receives ZoneAgent3s pro-

posal with the power available, it sends a proposal to ZoneAgent1 using a PROPOSE message. ZoneAgent1 accepts the proposal of ZoneAgent2. Once this proposal is accepted, ZoneAgent2 sends a CONFIRM message to ZoneAgent3. The three agents perform the requested actions and send the control actions to their ServerAgents using a REQUEST message.

At t=3.5 sec., when a fault in Zone2 occurs, ServerAgent2 sends a REQUEST to ZoneAgent2 to inform the agent about the fault. ZoneAgent2 negotiates with its neighbors to reconfigure the system and to feed Zone1 since this is the zone with highest priority. ZoneAgent3 sends a proposal to ZoneAgent2, which is accepted and the system is reconfigured again. ZoneAgent2 initiates a negotiation process to feed its load, but it can not find helpers because Zone1 has a fault and Zone3 is serving Zone1's load. After the negotiation processes, each ZoneAgent sets the control signals and sends them to its ServerAgent using a REQUEST message.

At t=5.5 sec. when a fault in the SSCM of Zone1 is cleared, ServerAgent1 sends a REQUEST message to ZoneAgent1. ZoneAgent1 releases the resources sending a CANCEL message to ZoneAgent2. ZoneAgent2 is acting as a bridge, so it must send a CANCEL message to ZoneAgent3 who it is the owner of the resources released by ZoneAgent1. The ZoneAgents communicate to each other the new availability of resources with an INFORM_IF message. ZoneAgent2 initiates a negotiation process to find resources to feed its load. Both, ZoneAgent1 and ZoneAgent3 propose to ZoneAgent2. The system is reconfigured and can now feed the three loads.

At t=6.5 sec., when the fault in the SSCM of Zone2 is cleared, the system returns to its initial no-faults state . ZoneAgent2 releases the resources with a CANCEL message and the tie switches between the zones are opened again.

After the reconfiguration actions are decided, each ZoneAgent establishes the con-

trol signals and sends them to its ServerAgent, who transmits them to the Simulink™ model. Figure 5.3**a** shows the control signals for each zone; they are sent from ZoneAgents to ServerAgents. Each ServerAgent sends the signals through the socket and they are received by the client socket in the Simulink™ model of each zone. The Figure 5.3**a** shows the expected behavior based on the described fault conditions. In each Zone, the last control signal is used to connect or to disconnect the load and the other signals are used to control the connection with its neighbor zones.

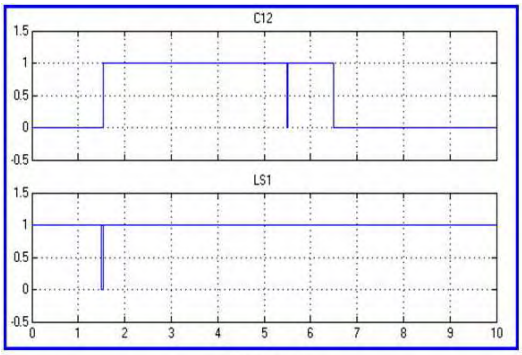
Figure 5.3**b** shows the voltage and current waveforms. They show how the MAS acts. Notice that when the fault occurred in Zone1, the voltage was affected because of the simulated communication delay of the control signals and the response time of the MAS, so the reconfiguration process allowed the system to feed the three loads.

5.2.1.2 Test with Different Priority Scheme

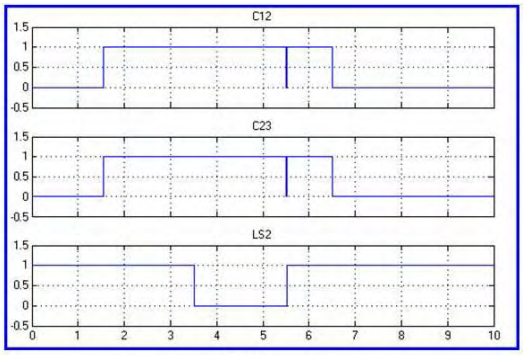
In this simulation scenario, the priorities of each zone are: Zone1 has the lowest priority 3, Zone2 has the highest priority 1, and Zone3 has priority 2. We keep the parameters of each scenario; we only change the priority scheme. Figures 5.4 and 5.5 show the Sniffer Agent GUI (Part 1 and Part 2).

The first part of the process operates in a similar way to the one presented above. But, when the second fault occurs (at t=3.5 seconds) in the SSCM of Zone2, the system can only feed one load, so the highest priority load is selected. With the new priority scheme, the highest priority is for Zone2. ZoneAgent3 sends a proposal to ZoneAgent2, which is accepted and the system is reconfigured to serve the load of Zone2. The last part of the process occurs in a similar way to the one presented with the default priority scheme.

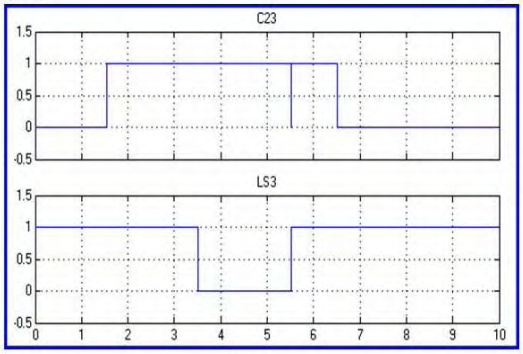
Figure 5.6**a** shows the control signals for each zone. Figure 5.6**b** shows the voltage and current waveforms. The MAS operates as it was expected, the reconfiguration process



(a) Zone 1

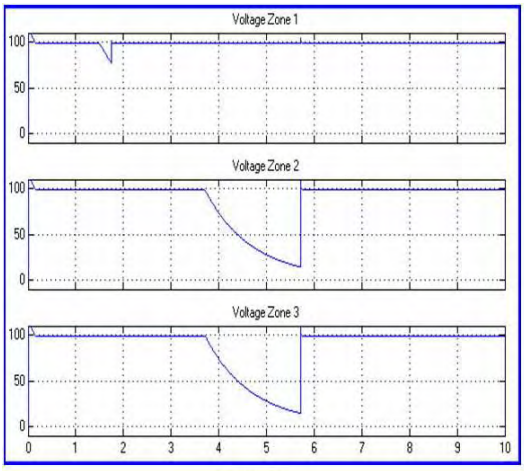


(b) Zone 2

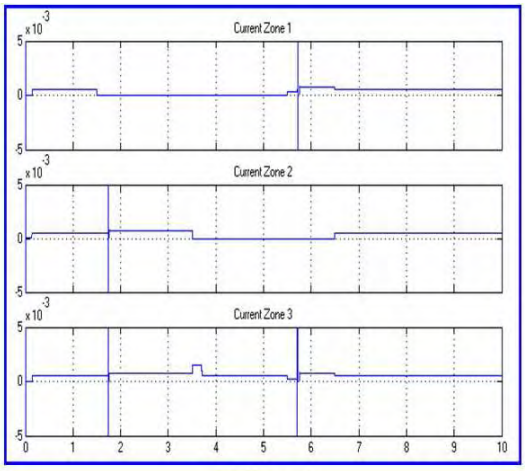


(c) Zone 3

(a) Time Diagram of Control Signals



(a) Voltage



(a) Current

(b) Zone Voltages and Currents.

Figure 5.3: BSB-DCZS Scenario I - Default Priority - Simulation Results.

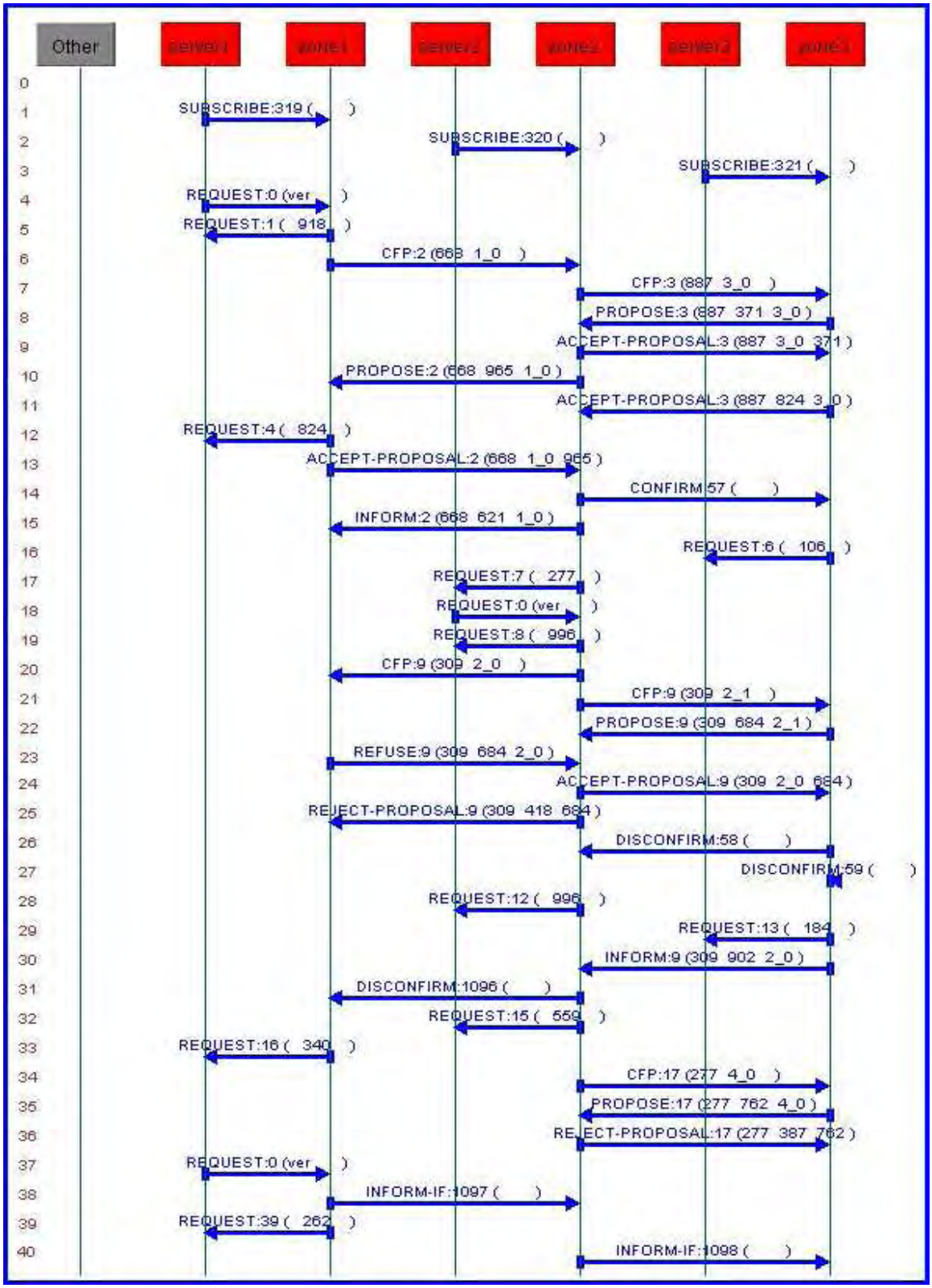


Figure 5.4: BSB-DCZS Scenario I - Different Priority: Sniffer Agent GUI - Negotiation Process (Part 1).

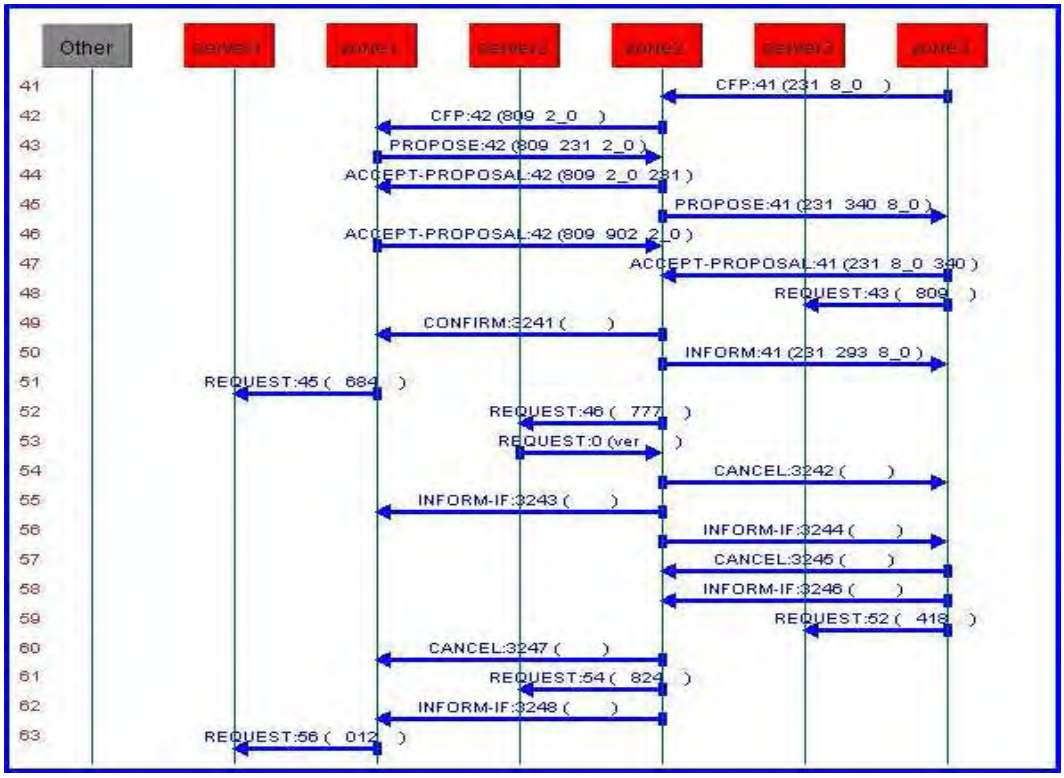


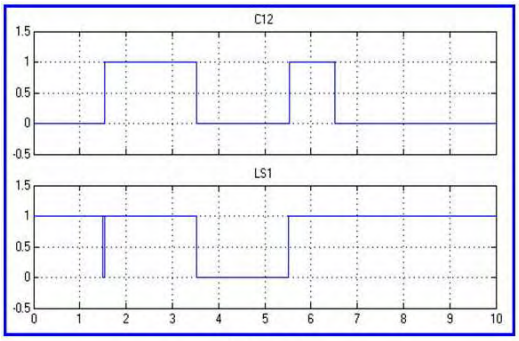
Figure 5.5: BSB-DCZS Scenario I - Different Priority: Sniffer Agent GUI - Negotiation Process (Part 2).

allowed the system to maximize the loads served with the highest priority.

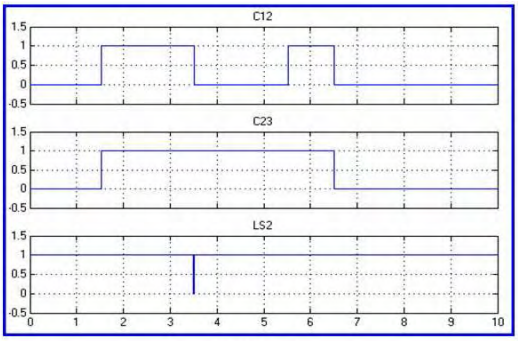
5.2.2 BSB-DCZS - Scenario II

The general parameters for Scenario II are:

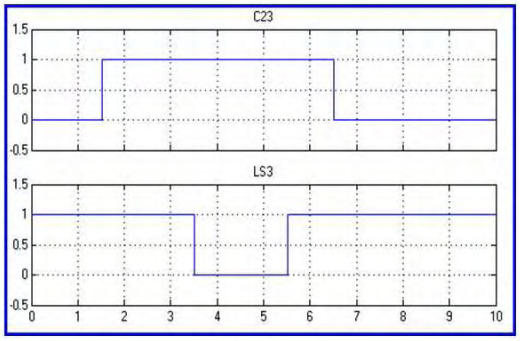
- Duration of Simulation: 10 seconds.
- At the beginning of the simulation the system is in normal state (no faults).
- At t=1.5 seconds, a fault takes place in the SSCM switch that feeds Zone2. The MAS must respond to this event and reconfigure the system, so the load of Zone2 is fed by the connection of the other two zones. It is necessary to close the switches between zones for this to occur.



(a) Zone 1

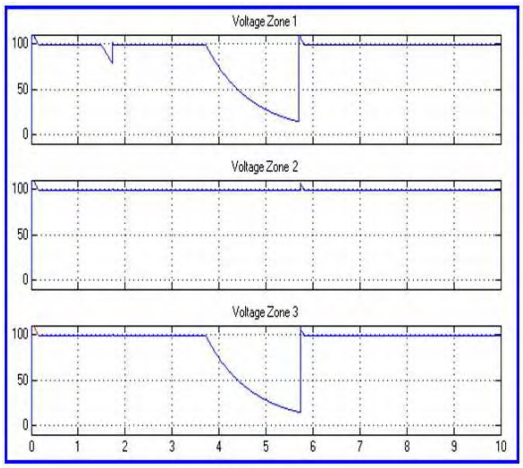


(b) Zone 2

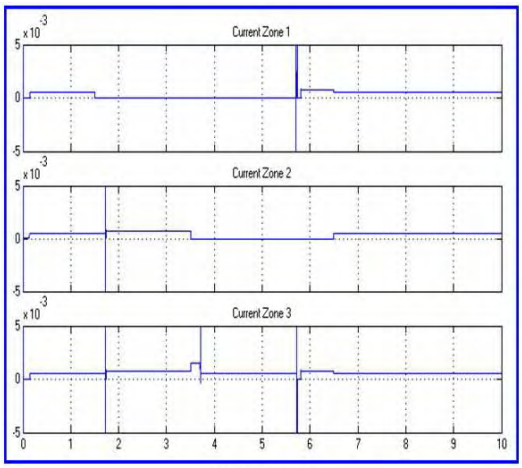


(c) Zone 3

(a) Time Diagram of Control Signals



(a) Voltage



(a) Current

(b) Zone Voltages and Currents.

Figure 5.6: BSB-DCZS Scenario I - Different Priority - Simulation Results.

- At t=3 seconds, the fault in the SSCM of Zone2 is cleared and the system is restored to its initial no-faults state.
- At t=4 seconds, a fault takes place in the SSCM of Zone1. The load of Zone1 is fed by the connection of the other two zones, closing the tie switches between zones.
- At t=6.5 seconds, a fault takes place in the SSCM of Zone3. At this moment, the system can feed only one Zone, so the highest priority Zone is selected.
- At t=8 seconds, the fault in the SSCM of Zone1 is cleared. The system can feed the three loads again. The load of Zone3 is fed by closing the tie switches between the zones.

5.2.2.1 Test with Default Priority Scheme

Figures 5.7 and 5.8 show the Sniffer Agent GUI (Part 1 and Part 2).

At the beginning of the simulation, the system is in normal state. When the fault in Zone2 occurs at t=1.5 sec., using the Contract Net Protocol, ZoneAgent2 negotiates with its neighbors ZoneAgent1 and ZoneAgent3 to reconfigure the system. After the negotiation, Zone1 and Zone3 provides Zone2 with enough resources to feed its load. The tie switches between zones are closed and the three loads are fed. At t=3, the fault in the SSCM of Zone2 is cleared, ZoneAgent2 sends a CANCEL message to release the resources of its neighbors and the system returns to its normal no faults state.

At t=4 sec., a fault in the SSCM of Zone1 occurs. ZoneAgent1 sends a CFP message to ZoneAgent2. ZoneAgent2 subcontracts resources to ZoneAgent3 and sends a proposal to ZoneAgent1. The proposal is accepted and the load of Zone1 is fed closing the tie switches between the zones.

At t=6.5 sec., when a fault in the SSCM of Zone3 occurs, the system only can feed one zone. The load of Zone1 is fed by the resources of Zone2. The connection between

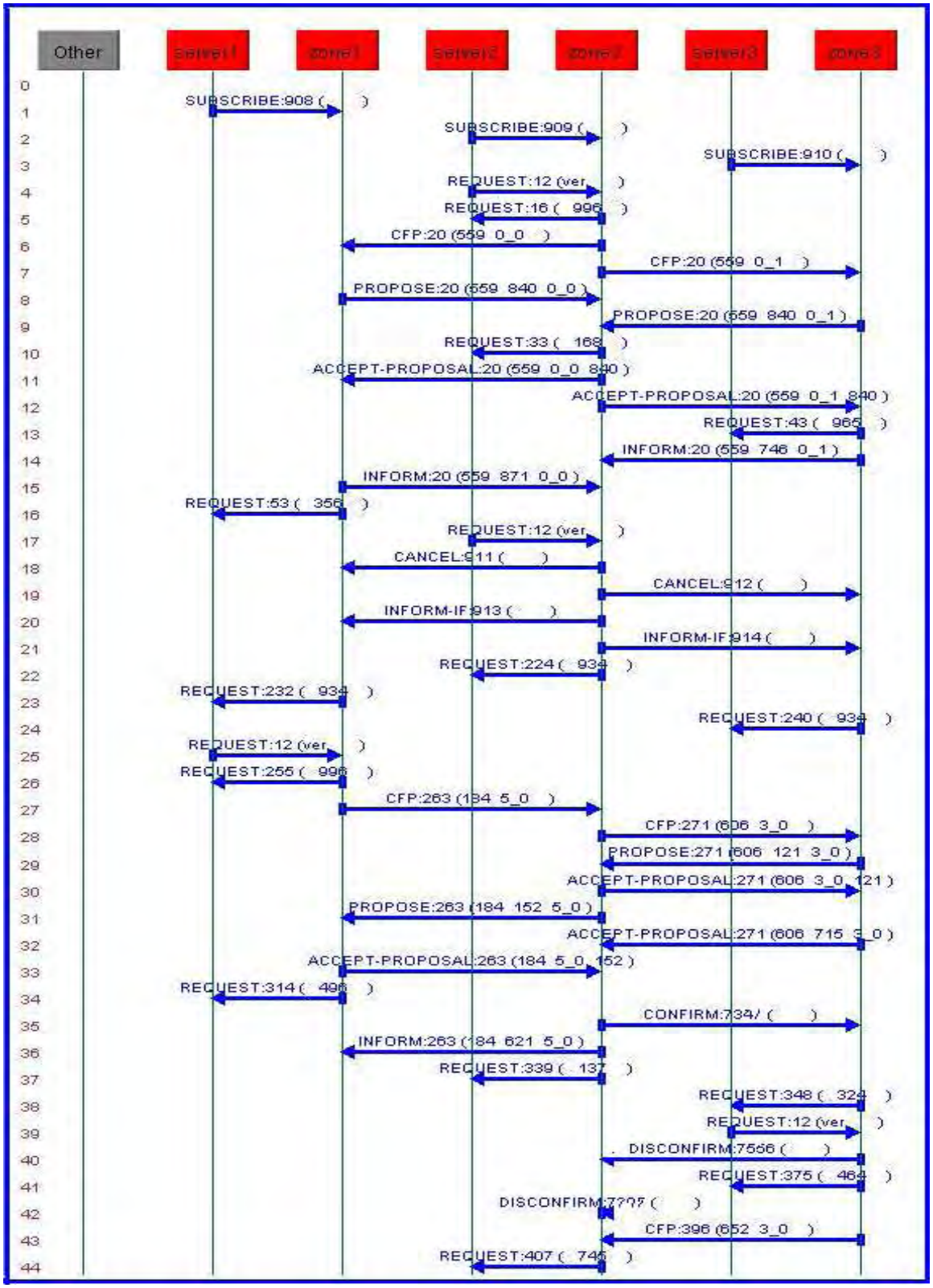


Figure 5.7: BSB-DCZS Scenario II - Default Priority: Sniffer Agent GUI - Negotiation Process (Part 1).

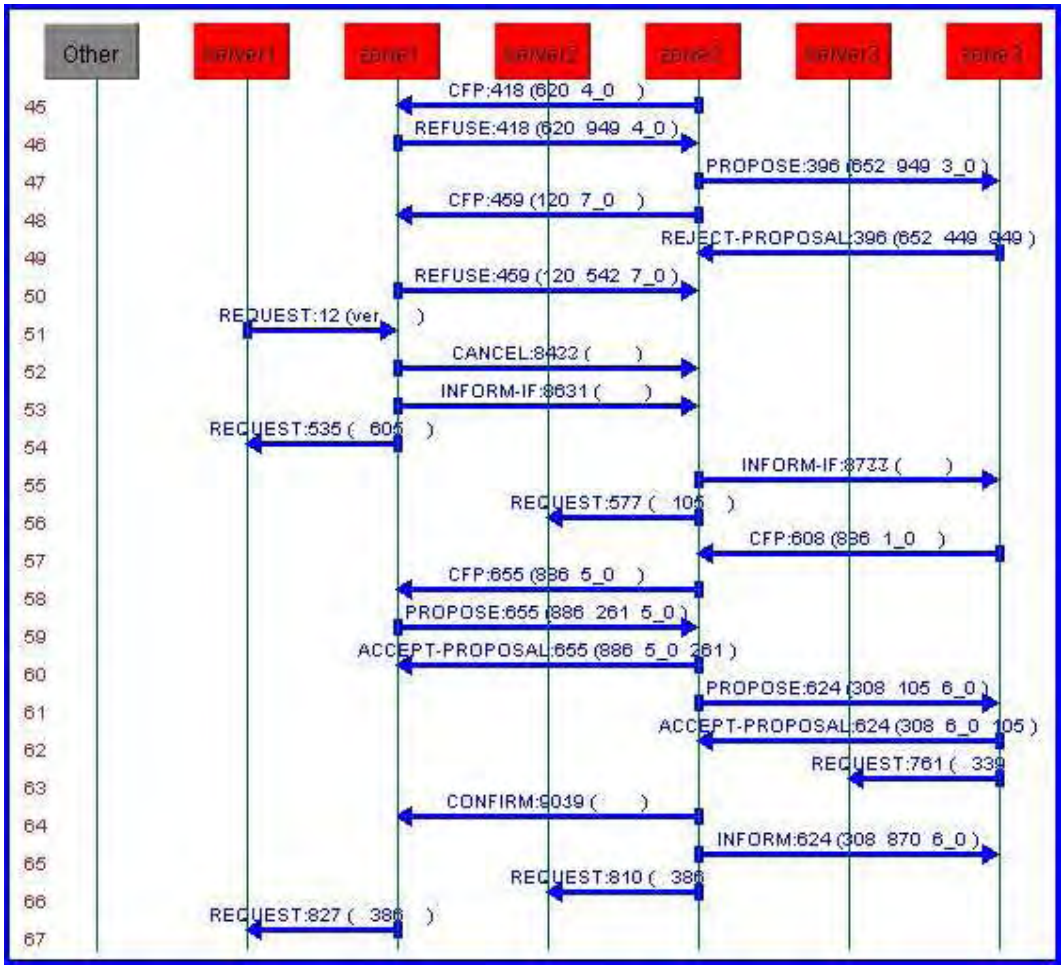
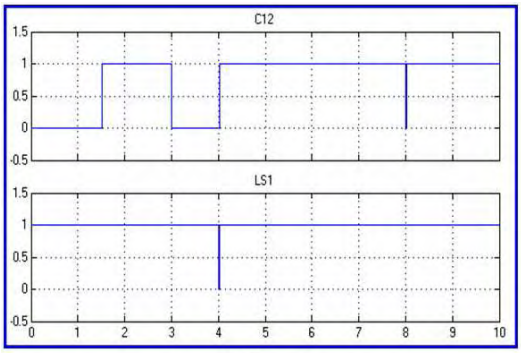


Figure 5.8: BSB-DCZS Scenario II - Default Priority: Sniffer Agent GUI - Negotiation Process (Part 2).

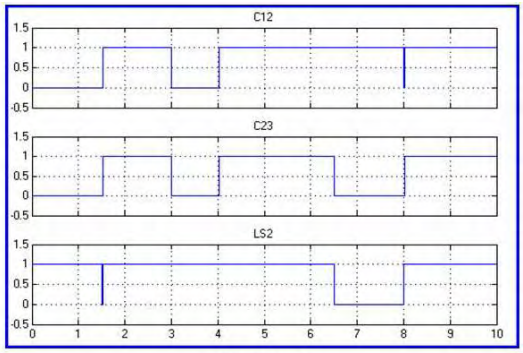
zones 1 and 2 is closed and the tie switch between Zone2 and 3 is opened.

At t=8 sec. when a fault in the SSCM of Zone1 is cleared, ZoneAgent1 releases the resources of Zone2. ZoneAgent2 connects its load and informs its neighbor ZoneAgent3 about this change; then a negotiation process between ZoneAgents starts in order to reconfigure the system to feed the three loads.

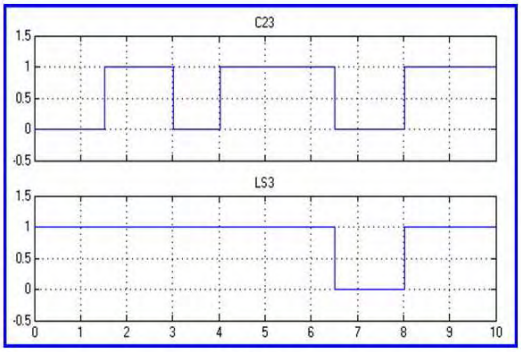
Figure 5.9**a** shows the control signals for each zone, and Figure 5.9**b** shows the voltage and current waveforms.



(a) Zone 1

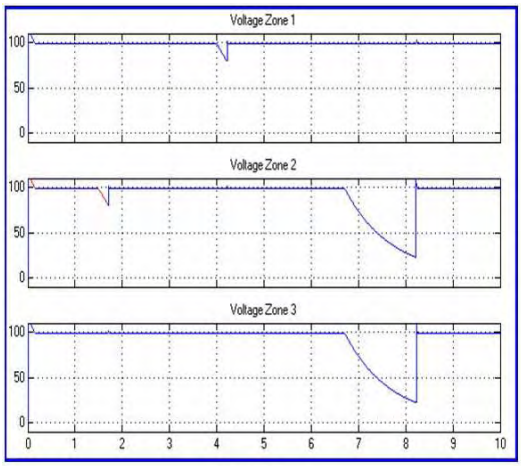


(b) Zone 2

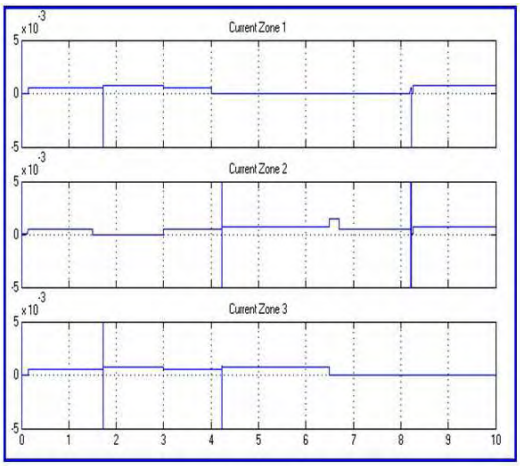


(c) Zone 3

(a) Time Diagram of Control Signals



(a) Voltage



(a) Current

(b) Zone Voltages and Currents.

Figure 5.9: BSB-DCZS Scenario II - Default Priority - Simulation Results.

5.2.2.2 Test with Different Priority Scheme

In this simulation scenario, the priorities of each zone are: Zone1 has priority 2, Zone2 has the lowest priority 3, and Zone3 has the highest priority 1. We keep the parameters of each scenario; we only change the priority scheme. Figures 5.10 and 5.11 show the Sniffer Agent GUI (Part 1 and Part 2). The first part of the process operates identical to the one presented above.

When a fault in the SSCM of Zone3 occurs (at t=6.5 seconds), the system can only feed one load, so the highest priority load is selected. With this new scheme, the highest priority is for Zone3. ZoneAgent2 disconnects its load and sends a proposal to ZoneAgent3, which is accepted and the system is reconfigured to serve the load of Zone3. The connection between Zone1 and Zone2 is opened, and the connection between Zone2 and Zone3 is closed. The last part of the process occurs identical to the one presented with the default priority scheme.

Figure 5.12**a** shows the control signals for each zone. Figure 5.12**b** shows the voltage and current waveforms.

5.3 Simulations results for BDB-DCZS

The second test system is the Basic Double Bus DC Zonal System Model (BDB-DCZS). As explained in Section 4.2.2, the BDB-DCZS is a simple circuit motivated by the dual bus architecture. The BDB-DCZS has three zones with identical RC loads. Each zone has two SSCMs, each one is connected to the DC source through a distribution bus.

5.3.1 BDB-DCZS - Scenario I

The general parameters for Scenario I are:

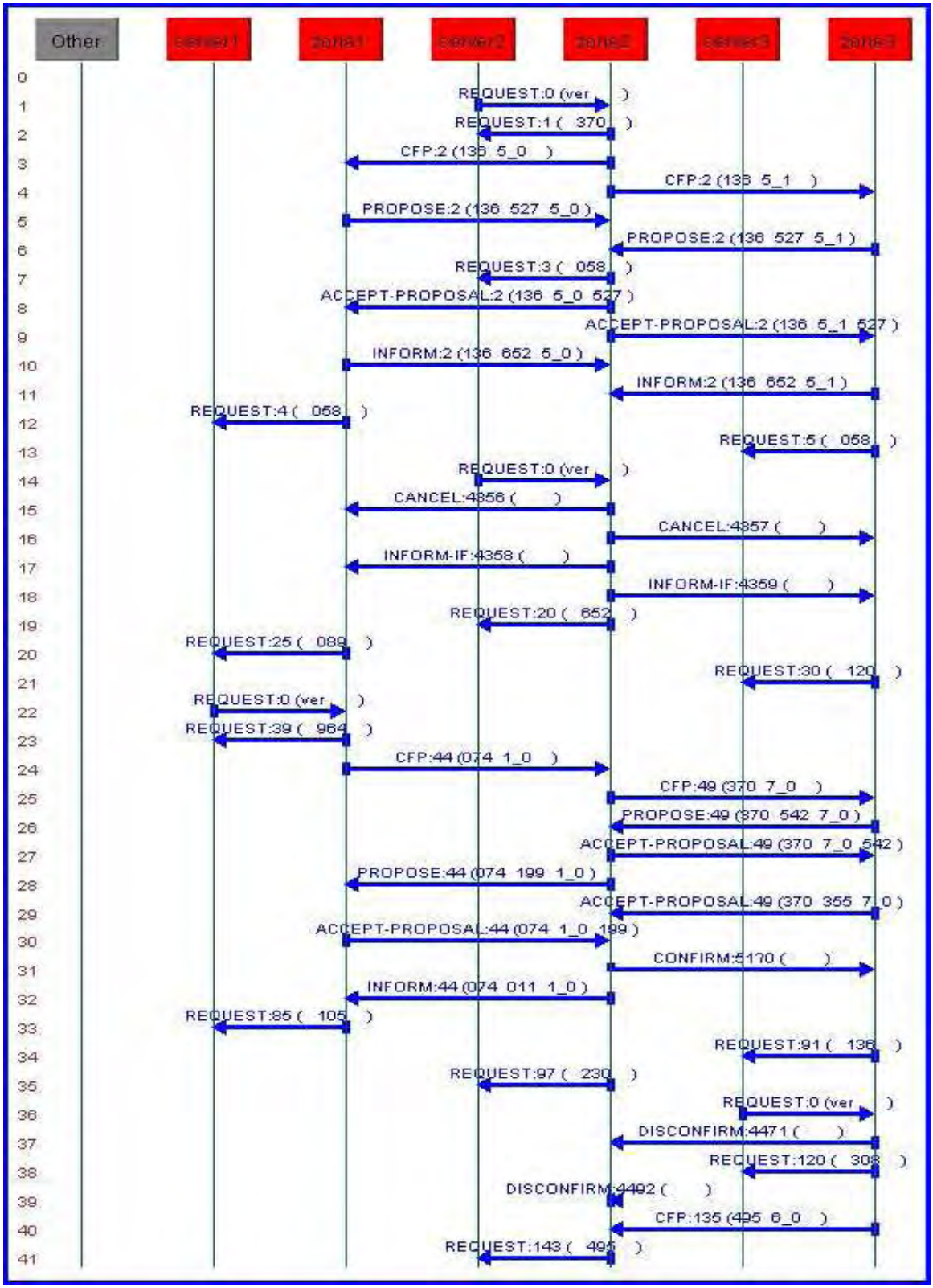


Figure 5.10: BSB-DCZS Scenario II - Different Priority: Sniffer Agent GUI - Negotiation Process (Part 1).

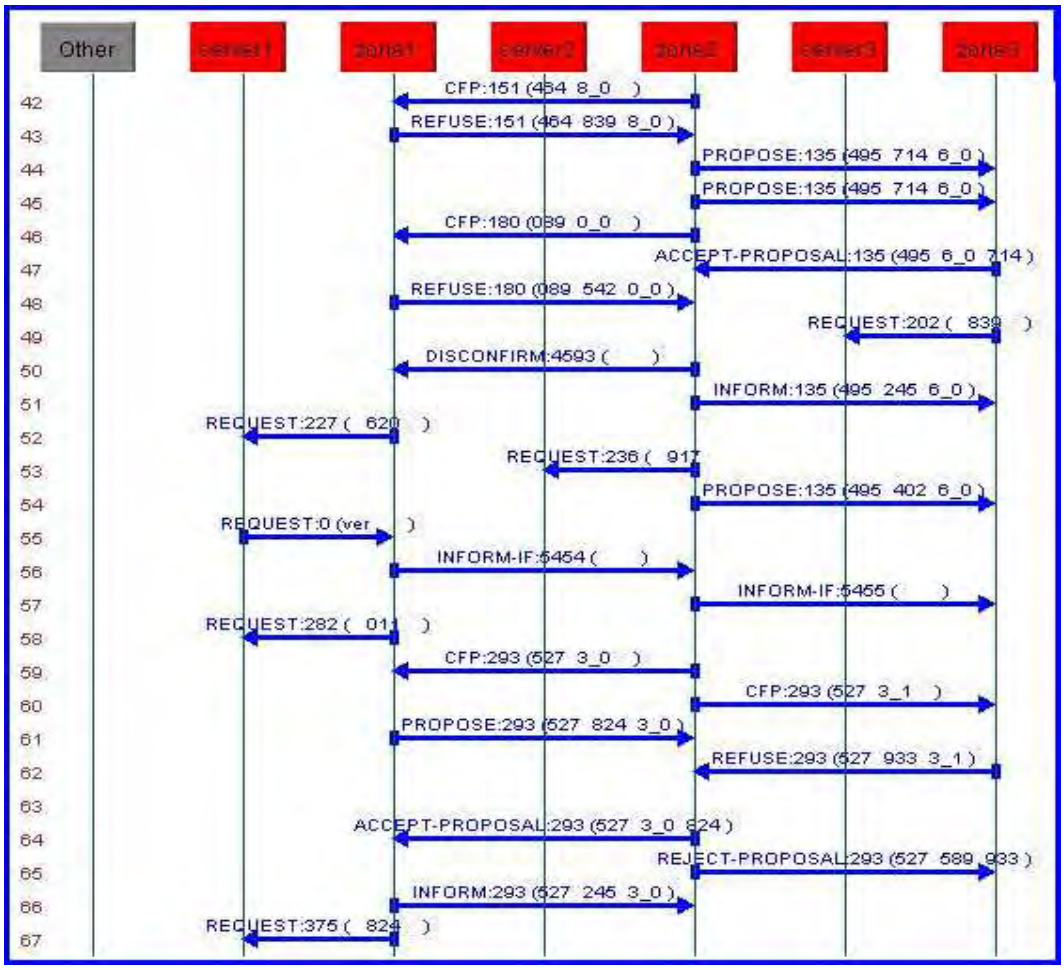
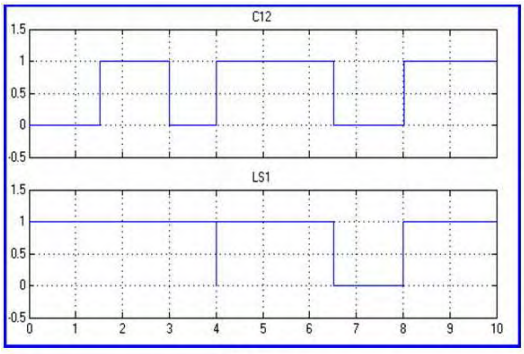
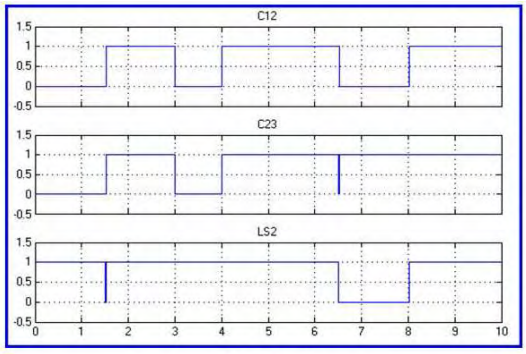


Figure 5.11: BSB-DCZS Scenario II - Different Priority: Sniffer Agent GUI - Negotiation Process (Part 2).

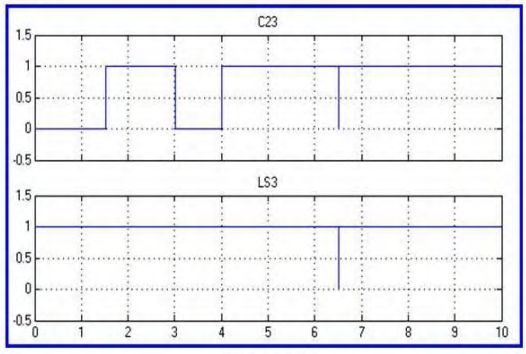
- Duration of Simulation: 10 seconds.
- At the beginning of the simulation the system is in normal state (no faults).
- At t=1 second, a fault takes place in the SSCM of Zone3, connected to the Port Bus. The MAS must respond to this event and reconfigure the system. Zone3 has the SSCM connected to the Starboard bus available, so the connection is changed and the load is fed using the resources of the Starboard bus.
- At t=3 seconds, a fault takes place in the SSCM of Zone3, connected to the Starboard bus. At this moment, Zone3 cannot feed its load. The load of Zone3 is fed by the



(a) Zone 1

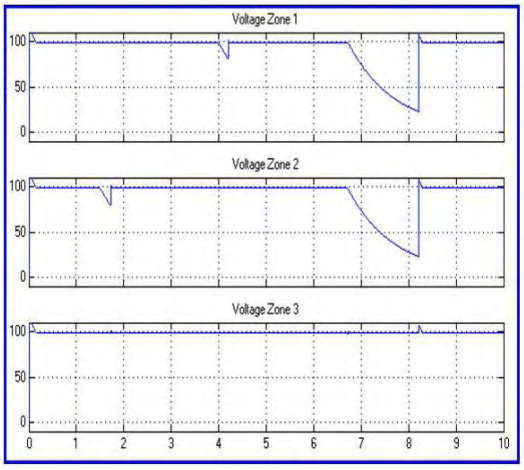


(b) Zone 2

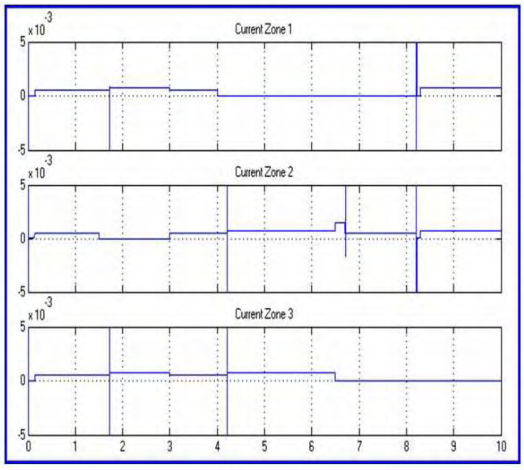


(c) Zone 3

(a) Time Diagram of Control Signals



(a) Voltage



(a) Current

(b) Zone Voltages and Currents.

Figure 5.12: BSB-DCZS Scenario II - Different Priority - Simulation Results.

connection of the other two zones.

- At t=5 seconds, a fault takes place in the SSCM of Zone2 connected to the Port Bus. ZoneAgent2 must move its clients to the connection to the Starboard bus. The system can now feed the three loads.
- At t=6.5 seconds, the SSCM of Zone2 connected to the Starboard bus has a fault. At this point, only the SSCM of Zone1 is available. Based on the constrains of the system, only one load can be fed for each SSCM, so the loads of Zone1 and Zone2 are fed, because they have the highest priority.

5.3.1.1 Test with Default Priority Scheme

For the first simulation of this scenario, we use the default priority scheme. Figure 5.13 shows the Sniffer Agent GUI.

When the system starts, each connection between the zones in the Simulink™ model and their ServerAgents in the MAS is open, and each ServerAgent sends a SUBSCRIBE message to its ZoneAgent. During the simulation the client socket in the Simulink™ Model sends information regarding the status of the SSCM to the ServerAgent.

When a fault in Zone3 occurs at t=1 sec., the ServerAgent3 sends a REQUEST message to ZoneAgent3 to inform the agent about the fault. The zone has the connection to the Starboard bus available, so the ZoneAgent3 decides to feed the load with the SSCM connected to the Starboard bus.

At t=3 sec., a fault takes place in the SSCM of Zone3 connected to the Starboard Bus. Zone3 has the two SSCMs in fault condition. ZoneAgent3, in representation of its zone, sends a CFP message to its neighbor ZoneAgent2. ZoneAgent2 checks the available resources of its zone, but does not have enough to feed both zones. ZoneAgent2 needs to obtain the other part of the power requested by ZoneAgent3, so, it sends a CFP to its neighbor

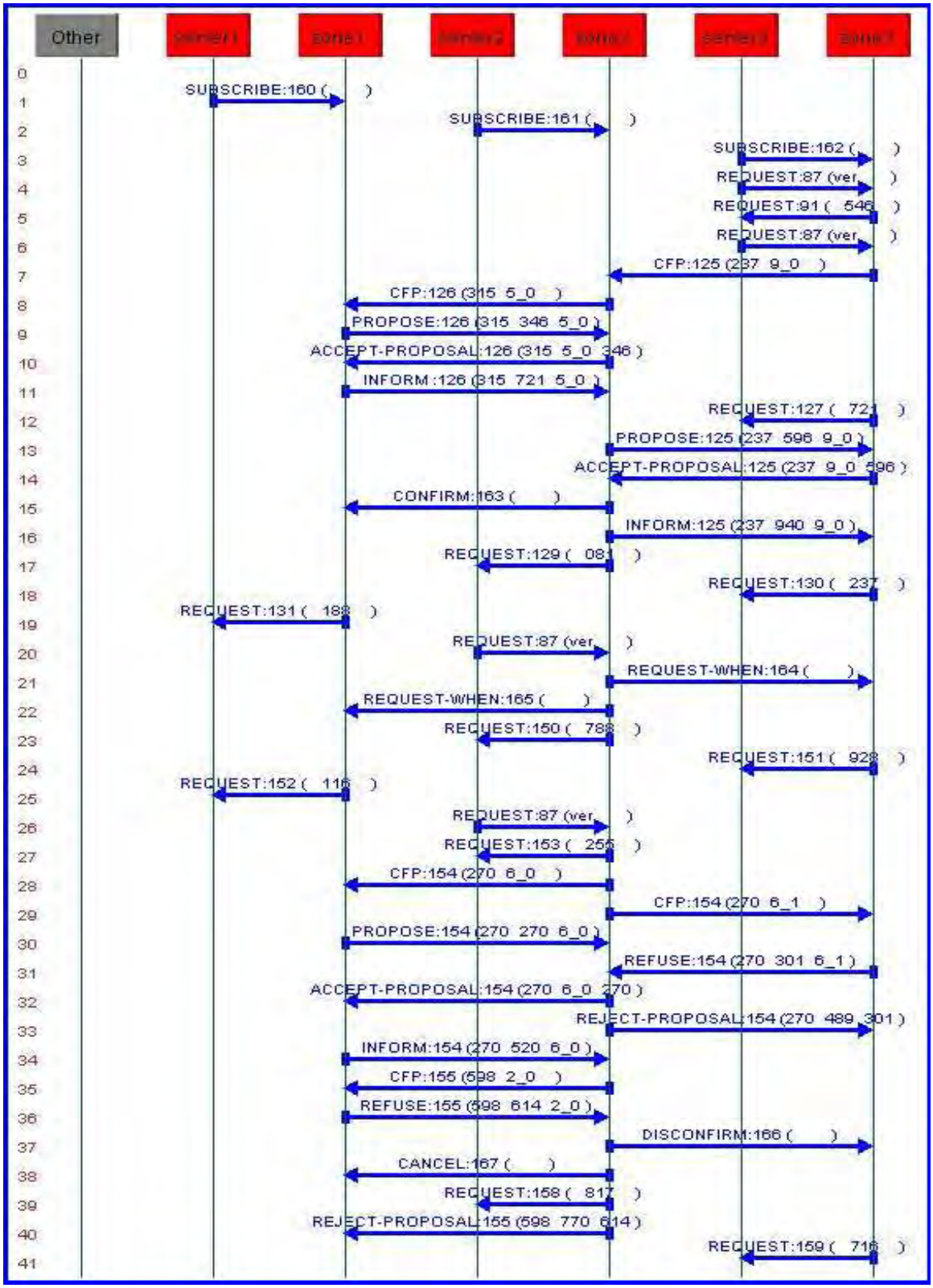


Figure 5.13: BDB-DCZS Scenario I - Default Priority: Sniffer Agent GUI - Negotiation Process.

ZoneAgent1. According to Contract Net Protocol, ZoneAgent1 sends a PROPOSE message to ZoneAgent2, this proposal is accepted by ZoneAgent2 using an ACCEPT-PROPOSAL message. When ZoneAgent2 receives the proposal by ZoneAgent1 and the power available, it sends a proposal to ZoneAgent3 using a PROPOSE message. ZoneAgent3 accepts the proposal of ZoneAgent2. When this proposal is accepted, it sends a CONFIRM message to ZoneAgent1. The three agents requests actions and send the control actions to their ServerAgents using a REQUEST message.

When a fault in the SSCM of Zone2 connected to the Port bus occurs at t=5 sec., ZoneAgent2 needs to move its clients to the connection with the Starboard Bus. ZoneAgent2 has two clients, ZoneAgent3 and itself. The movement of its load to the Starboard bus is not a problem, but to move ZoneAgent3's load, it needs to communicate with ZoneAgent1 because part of ZoneAgent3's resources were subcontracted to this agent. ZoneAgent2 coordinates this change with its neighbors, using REQUEST_WHEN messages. The system still has the capacity to feed the three loads.

At t=6.5 sec., a fault occurs in the SSCM of Zone2 connected to the Starboard bus. At this point, only the SSCM of Zone1 is available. Based on the constrains of the system, only one load can be fed for each SSCM, so the loads of Zone1 and Zone2 are fed, because they have the highest priority. ZoneAgent2 sends CFP messages to ZoneAgent1 and ZoneAgent2 to feed its load and the load of Zone3 (ZoneAgent2 has ZoneAgent3 as a client). ZoneAgent3 can not help because it does not have resources available. Based on the priority scheme, ZoneAgent1 sends a proposal to feed the load of Zone2 and a REFUSE message to ZoneAgent3. ZoneAgent2 sends a DISCONFIRM message to ZoneAgent3 to inform that it can not receive the resources to feed its load. Also, ZoneAgent2 sends a CANCEL message to ZoneAgent1 to release the resources that were being used to help the Zone3. The load of Zone1 is fed by the SSCM of Zone1 connected to the Port bus and the load in Zone2 is fed by the SSCM of Zone1 connected to the Starboard bus.

After the reconfiguration actions are decided, each ZoneAgent establishes its control signals which it sends to its ServerAgent, who transmits them to the Simulink™ model. Figure 5.14**a** shows the control signals for each zone; they are sent from ZoneAgents to ServerAgents. Each ServerAgent applies the signals to the electrical model using the communication middleware. The Figure shows the expected behavior based on the described fault conditions. Figure 5.14**b** shows the voltage and current waveforms.

5.3.1.2 Test with Different Priority Scheme

In this simulation scenario, the priorities of each zone are: Zone1 has the lowest priority 3, Zone2 has the highest priority 1, and Zone3 has priority 2. We keep the parameters of each scenario; we only change the priority scheme. Figure 5.15 shows the Sniffer Agent GUI. The first part of the process operates in the same way as the one presented previously.

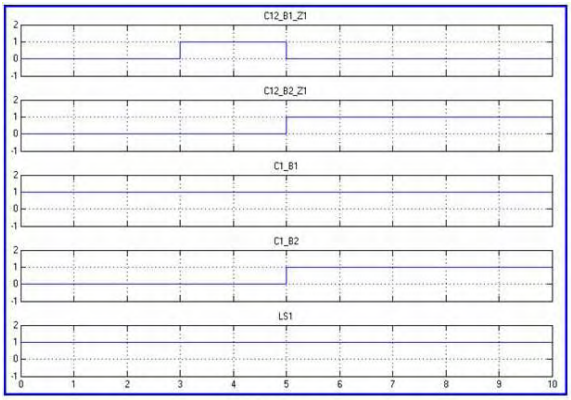
When a fault in the SSCM of Zone2 connected to the Starboard bus occurs (at t=6.5 seconds), the system only can feed two loads, so the highest priority loads are selected. With the new priority scheme, the highest priority zones are Zone2 and Zone3. ZoneAgent2 negotiates with ZoneAgent1 and Zone2 and Zone3 loads are served by connections of Zone1 to the Port Bus and the Starboard bus.

Figure 5.16**a** shows the control signals for each zone. Figure 5.16**b** shows the voltage and current waveforms.

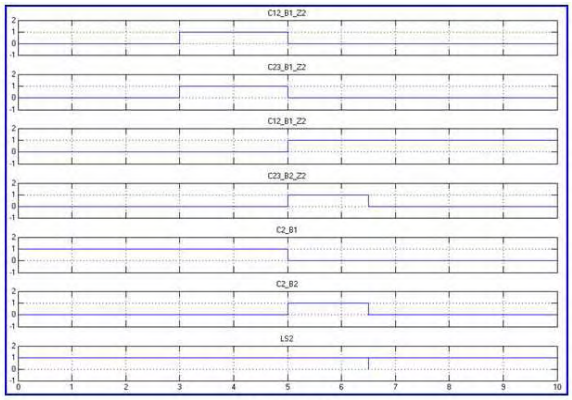
5.3.2 BDB-DCZS - Scenario II

The general parameters for Scenario II are:

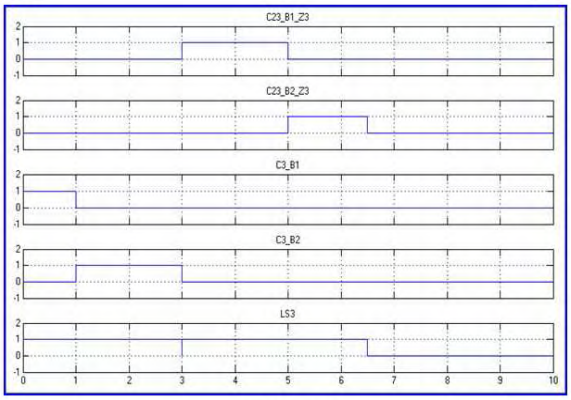
- Duration of Simulation: 10 seconds.
- At the beginning of the simulation the system is in normal state (no faults).



(a) Zone 1

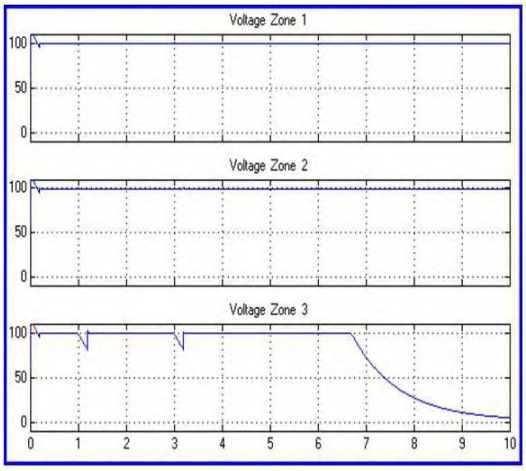


(b) Zone 2

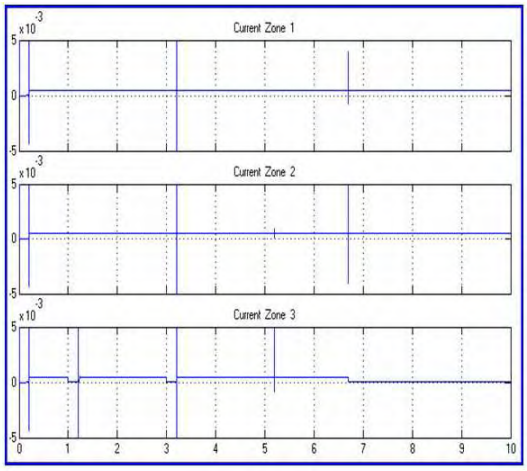


(c) Zone 3

(a) Time Diagram of Control Signals



(a) Voltage



(a) Current

(b) Zone Voltages and Currents.

Figure 5.14: BDB-DCZS Scenario I - Default Priority - Simulation Results.

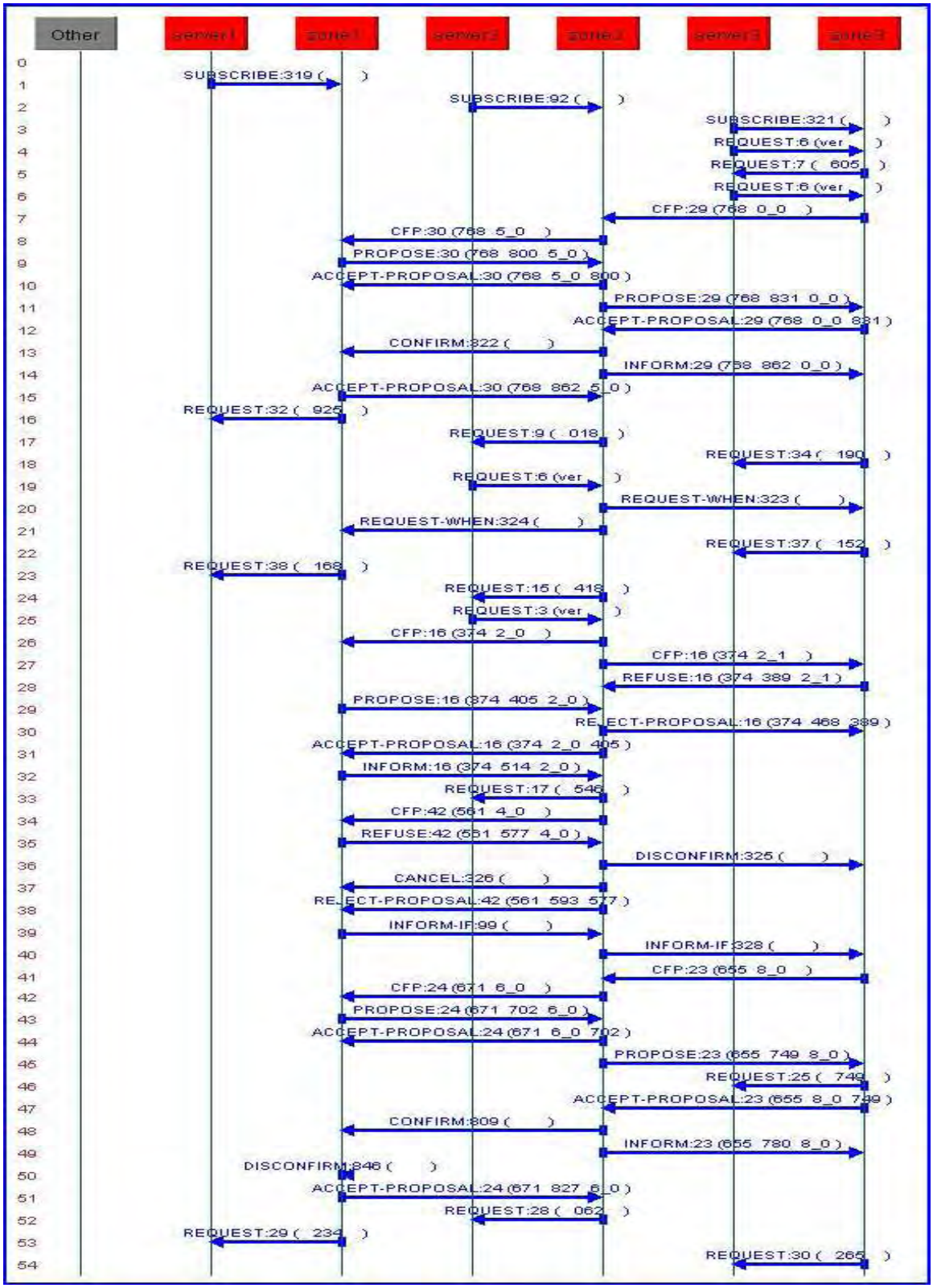
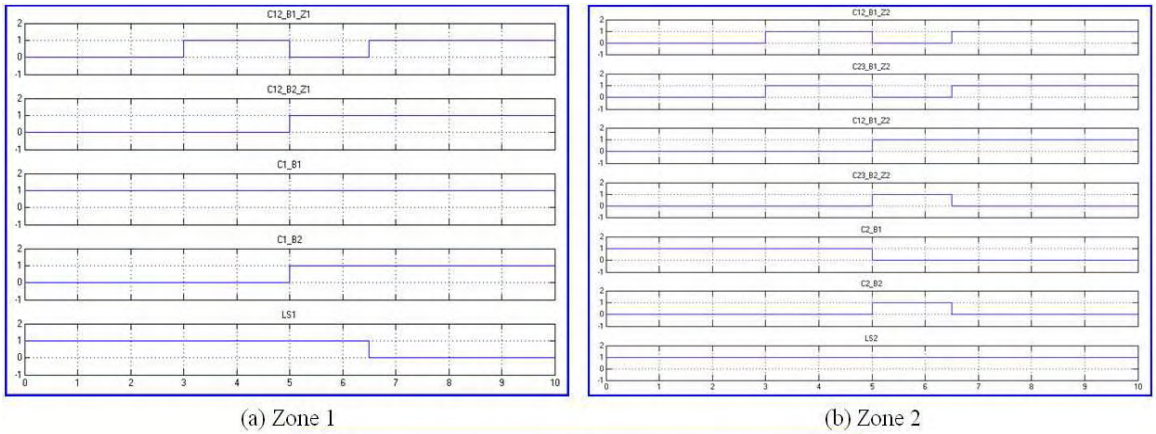
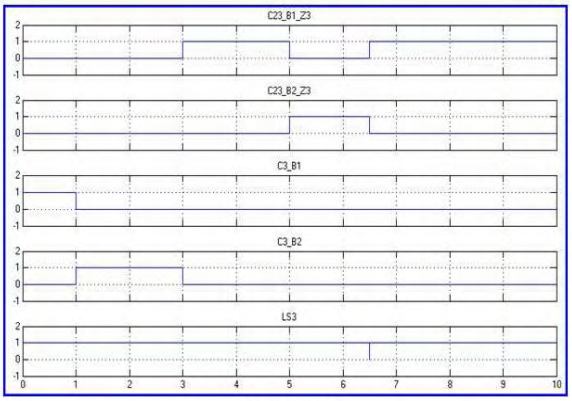


Figure 5.15: BDB-DCZS Scenario I - Different Priority: Sniffer Agent GUI - Negotiation Process.

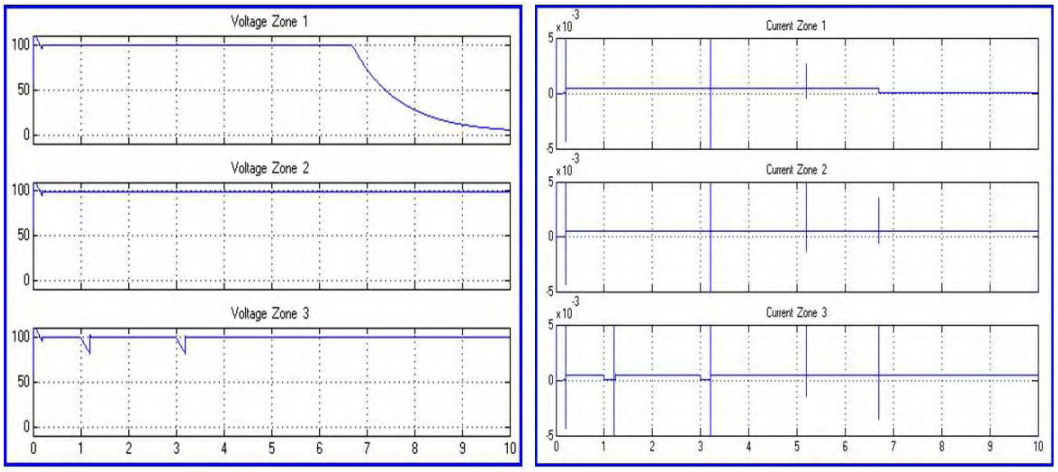


(a) Zone 1 (b) Zone 2



(c) Zone 3

(a) Time Diagram of Control Signals



(a) Voltage (a) Current

(b) Zone Voltages and Currents.

Figure 5.16: BDB-DCZS Scenario I - Different Priority - Simulation Results.

- At t=1 second, a fault takes place in the SSCM of Zone1, connected to the Port Bus. The MAS must respond to this event and reconfigure the system. Zone1 has the SSCM connected to the Starboard bus available, so the connection is changed and the load is fed using the resources of the Starboard bus.
- At t=2.5 seconds, a fault takes place in the SSCM of Zone3, connected to the Port Bus. Zone3 has the SSCM connected to the Starboard bus available, so the connection is changed and the load is fed using the resources of the Starboard bus.
- At t=4 seconds, a fault takes place in the SSCM of Zone1, connected to the Starboard bus. At this moment, Zone1 can not feed its load. The load of Zone1 is fed by the connection of Zone2 to the Starboard Bus.
- At t=5.5 seconds, a fault takes place in the SSCM of Zone2 connected to the Starboard Bus. The system only has two SSCMs available: the SSCM of Zone2 connected to the Port Bus and the SSCM of Zone3 connected to the Starboard Bus. Only two loads can be fed and they are selected based on the priority scheme.
- At t=7 seconds, a fault in the SSCM of Zone1 connected to the Port bus is cleared. Zone1 releases the resources and each zone can connect its load using the SSCM available.
- At t=8.5 seconds, a fault in the SSCM of Zone3 connected to the Port bus is cleared. Zone3 connects its zone to the Port bus. All loads are fed by the SSCM connected to the Port bus.

5.3.2.1 Test with Default Priority Scheme

For the first simulation, we use the default priority scheme. Figure 5.17 shows the Sniffer Agent GUI.

At the beginning of the simulation the system is in normal state. When a fault occurs in the SSCM of Zone1 connected to the Port Bus at t=1 sec., the ServerAgent1 sends

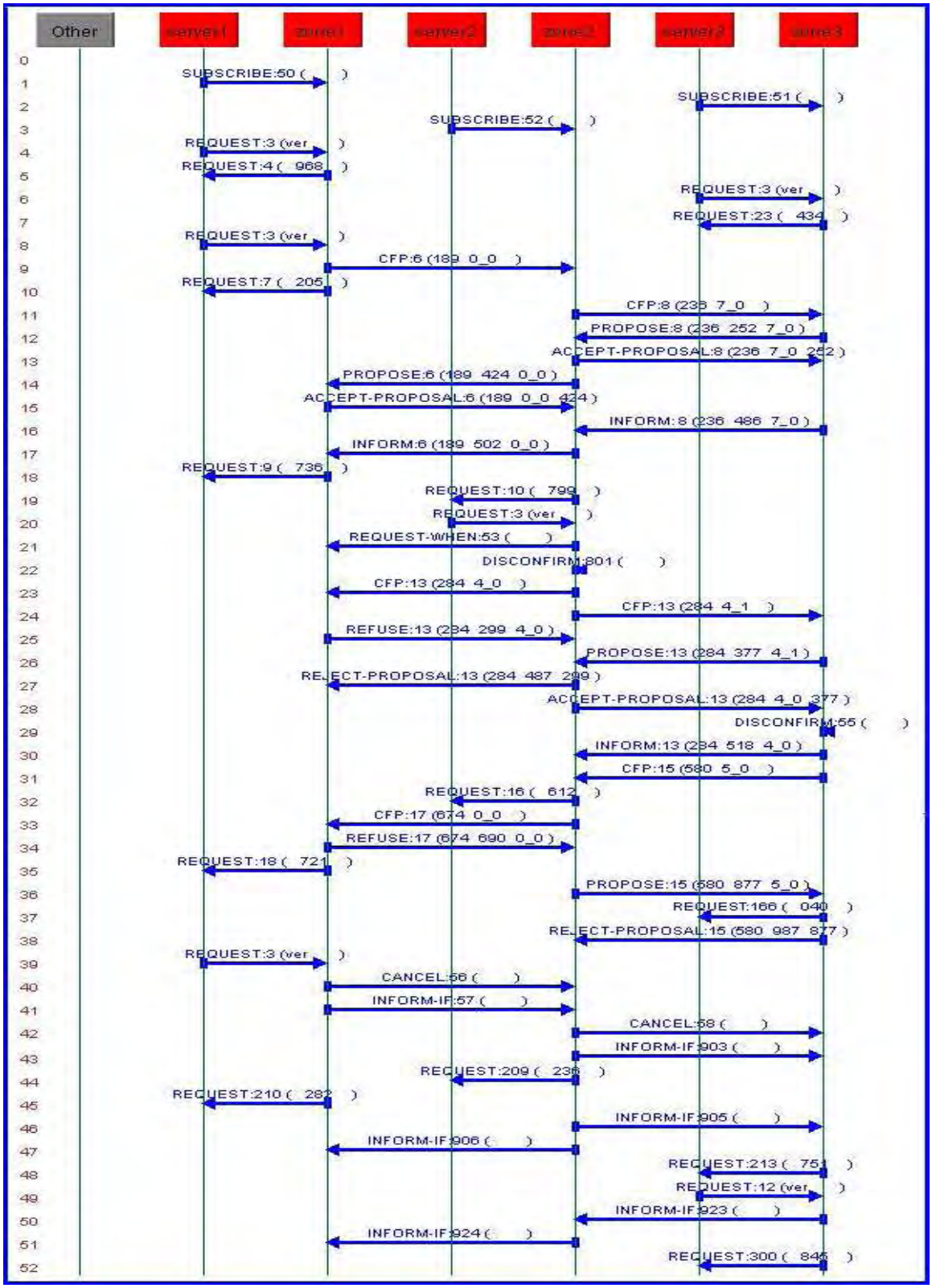


Figure 5.17: BDB-DCZS Scenario II - Default Priority: Sniffer Agent GUI - Negotiation Process.

a REQUEST message to ZoneAgent1 to inform the agent about the fault. The zone has the connection to the Starboard bus available, so the ZoneAgent1 decides to feed the load with the SSCM connected to the Starboard bus. The same process occurs when a fault occurs in the SSCM of Zone 3 connected to the Port bus at t=2.5 seconds.

At t=4 sec., a fault takes place in the SSCM of Zone1, connected to the Starboard bus. At this moment, Zone1 cannot feed its load. ZoneAgent1 sends a CFP message to ZoneAgent2. ZoneAgent2 sends a CFP message to ZoneAgent3, according to Contract Net Protocol. ZoneAgent1 accepts the ZoneAgent2's proposal to connect its load to the Starboard bus using the SSCM of Zone2.

At t=5.5 seconds, when a fault takes place in the SSCM of Zone2 connected to the Starboard Bus, the system only has two SSCMs available: the SSCM of Zone2 connected to the Port Bus and the SSCM of Zone3 connected to the Starboard Bus. Only two loads can be fed and they are selected based on the priority scheme. When ZoneAgent2 receives the notification about the fault from ServerAgent2, it must search for alternative resources to its clients based on the priorities. ZoneAgent1 is a client of the Starboard bus, and its zone is a client for the Port Bus. Because Zone1 has higher priority than Zone2; ZoneAgent2 interacts with ZoneAgent1 to connect the load of Zone1 using the SSCM connected to the Port Bus. Also, ZoneAgent2 disconnects its load to serve the load of Zone1. After ZoneAgent2 serves the client with the highest priority, it sends a CFP message to its neighbors in order to find resources for its load. ZoneAgent3 sends a proposal based on the priority scheme. ZoneAgent2 accepts the proposal and ZoneAgent3 disconnects its load to serve the load of Zone2.

When a fault in the SSCM of Zone1 connected to the Port Bus is cleared (at t=7 sec.), ZoneAgent1 connects its load to the Port Bus and sends a CANCEL message to releases the resources of Zone2. With the new availability of resources, ZoneAgent2

connects its load to the Port Bus and sends a CANCEL message to ZoneAgent3 to release resources. ZoneAgent3 connects its load to the Starboard Bus.

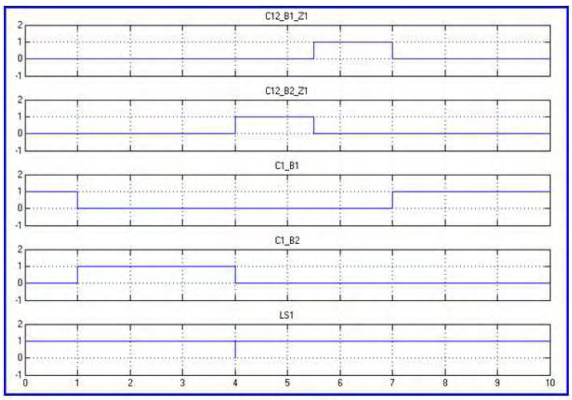
At t=8.5 sec., a fault in the SSCM of Zone3 connected to the Port Bus is cleared. ZoneAgent3 connects its load to the Port Bus. The three loads are now fed by their connections to the Port Bus.

Figure 5.18**a** shows the control signals for each zone. Figure 5.18**b** shows the voltage and current waveforms.

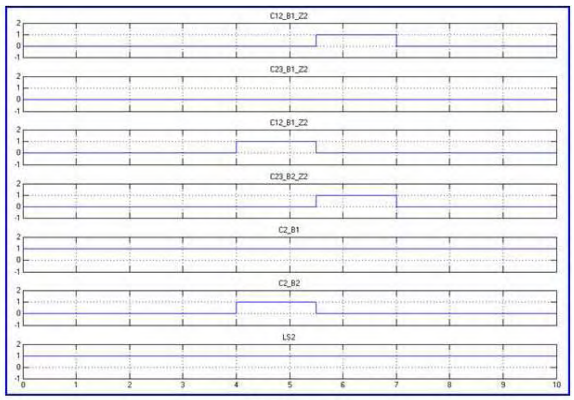
5.3.2.2 Test with Different Priority Scheme

Here, Zone1 has priority 2, Zone2 has the lowest priority 3, and Zone3 has the highest priority 1. We keep the parameters of each scenario; we only change the priority scheme. Figure 5.19 shows the Sniffer Agent GUI.

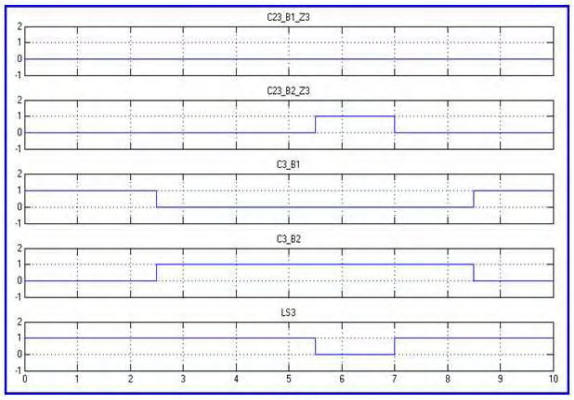
The first part of the process operates in the same way as the one presented previously. But, when a fault in the SSCM of Zone2 connected to the Starboard Bus occurs (at t=5.5 seconds), the system only can feed two loads, so the highest priority loads are selected. With the new priority scheme, the highest priority zones are Zone1 and zone 3. When ZoneAgent2 receives the notification regarding the fault from ServerAgent2, it has ZoneAgent1 as a client, connected to the Starboard bus. The priority of Zone1 is higher priority than Zone2, then, ZoneAgent2 interacts with ZoneAgent1 to connect the load of Zone1 using the SSCM connected to the Port Bus. ZoneAgent2 also disconnects its load to serve the load of Zone1. After ZoneAgent2 serves the client with the highest priority, it sends a CFP message to its neighbors in order to find resources for its load. Zone3 has higher priority than Zone2, so ZoneAgent3 can propose enough resources to feed the load of Zone2. At this moment, the loads of Zone1 and 3 are connected and the load of Zone2 is disconnected.



(a) Zone 1

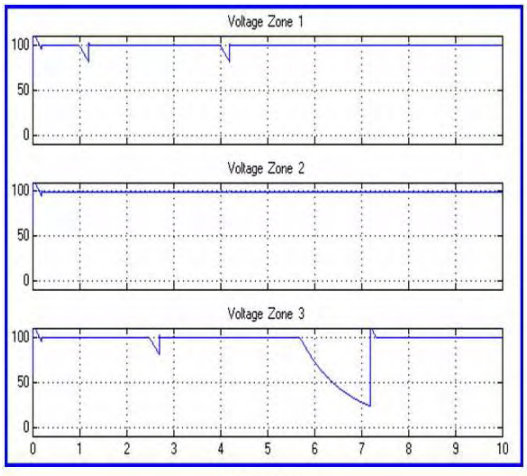


(b) Zone 2

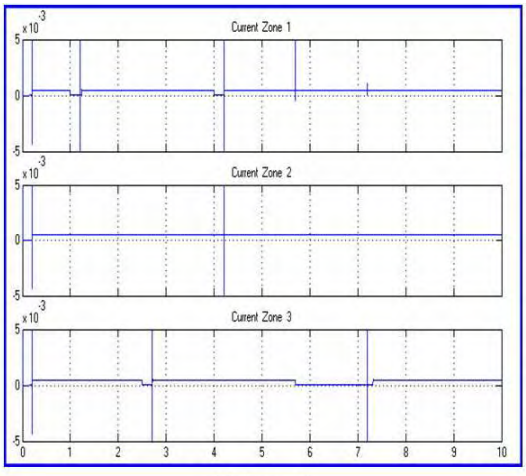


(c) Zone 3

(a) Time Diagram of Control Signals



(a) Voltage



(a) Current

(b) Zone Voltages and Currents.

Figure 5.18: BDB-DCZS Scenario II - Default Priority - Simulation Results.

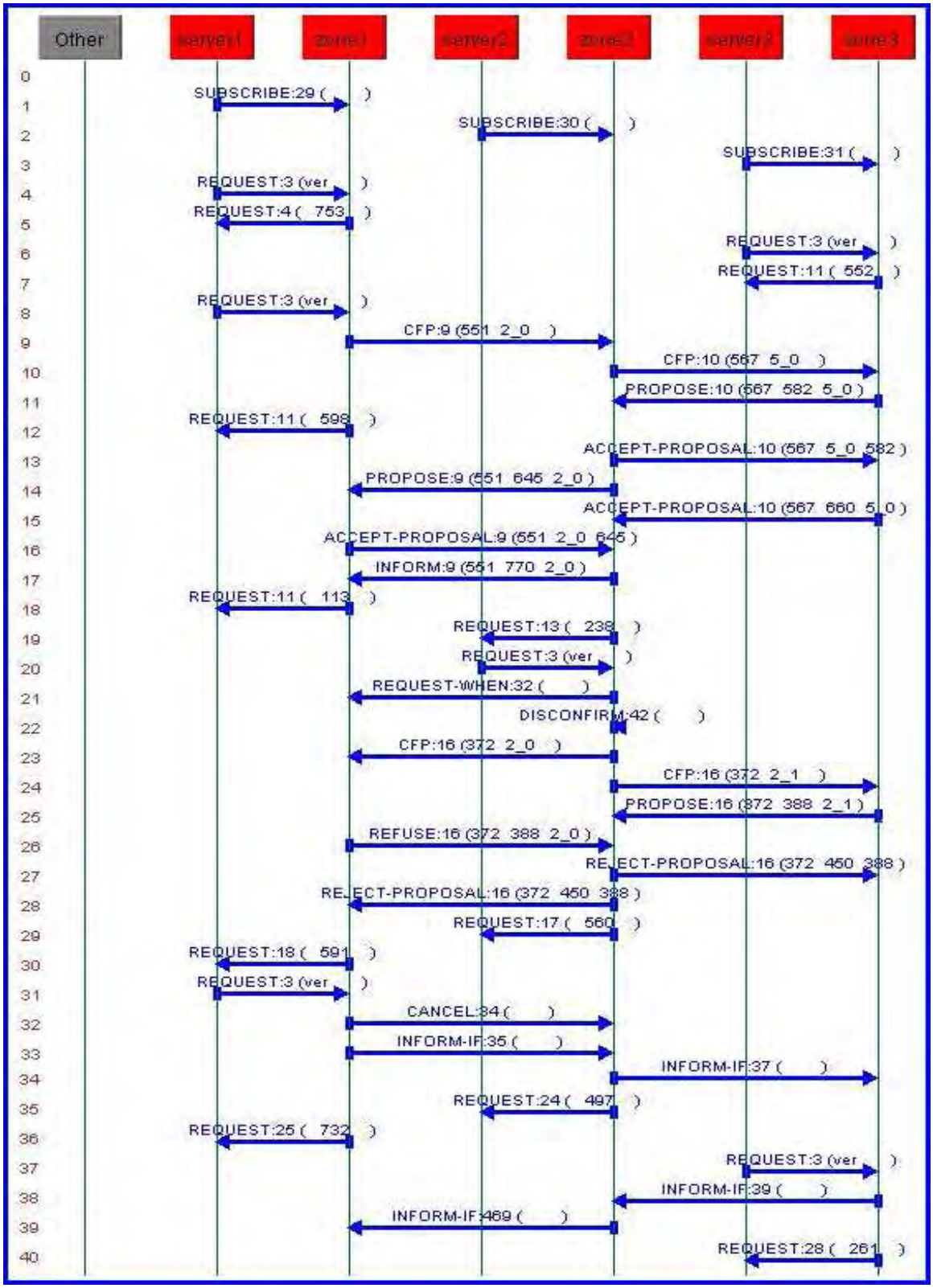


Figure 5.19: BDB-DCZS Scenario II - Different Priority: Sniffer Agent GUI - Negotiation Process.

The last part of the process occurs in the same way to the one presented with the default priority scheme. Figure 5.20**a** shows the control signals for each zone. Figure 5.20**b** shows the voltage and current waveforms.

5.4 Simulations results for RDB-DCZS

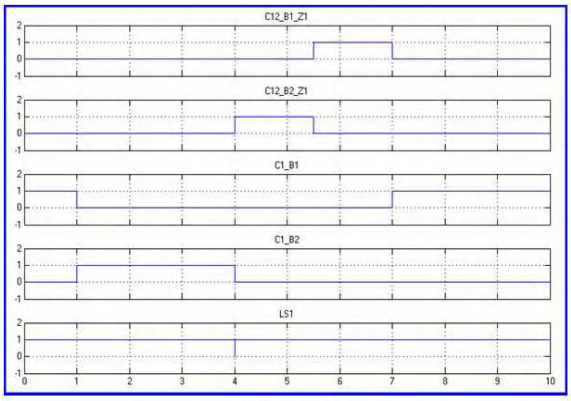
The third test system is the Realistic Dual Bus DC Zonal System Model (RDB-DCZS). As explained in Section 4.2.3, this model is based on the Simulink^{TM} implementation of the DC Zonal Electric Distribution System (DCZEDS) described on [18, 19, 20], provided by the Office of Naval Research (ONR) to researchers in the NSF/ONR EPNES program [20].

In this model, each SSCM accepts 500 V DC (this may vary), and regulates the output voltage to 400 V DC for load currents up to 20 A. Each load needs 5kW of power and a current of 12.5. Based on that information, a SSCM can supply up to 1.6 loads; with two SSCMs the system can feed the three loads. This constraint is similar to constraints of the previous simulation model.

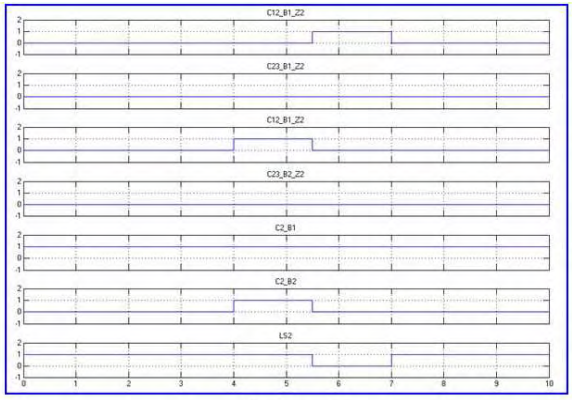
5.4.1 RDB-DCZS - Scenario I

The general parameters for Scenario I are:

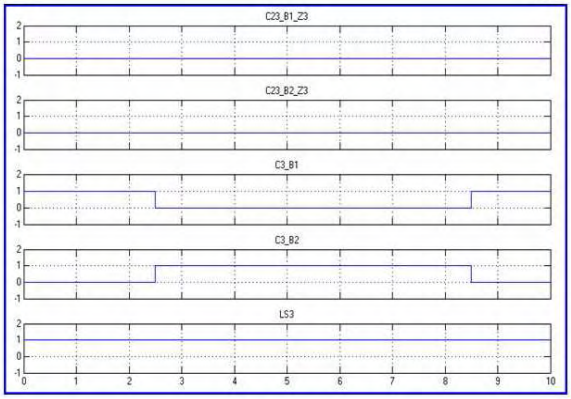
- Duration of Simulation: 1 second. The time of the simulation was fixed in order to facilitate visualization of the resulting figures, due to the presence of sinusoidal waveforms.
- At the beginning of the simulation the system is in normal state (no faults).
- At t=0.2 seconds, a fault takes place in the SSCM of Zone1, connected to the Starboard Bus. The MAS must store the information but no action is taken because the connection is not in use.



(a) Zone 1

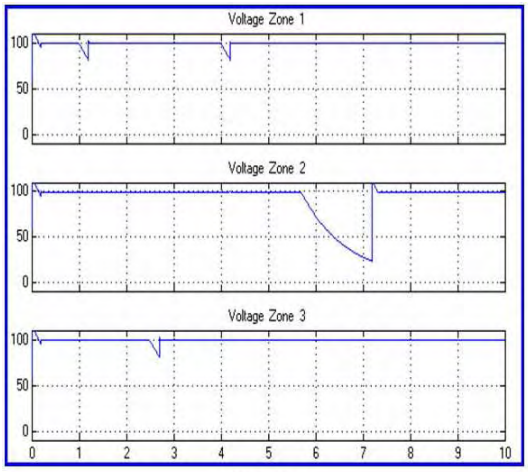


(b) Zone 2

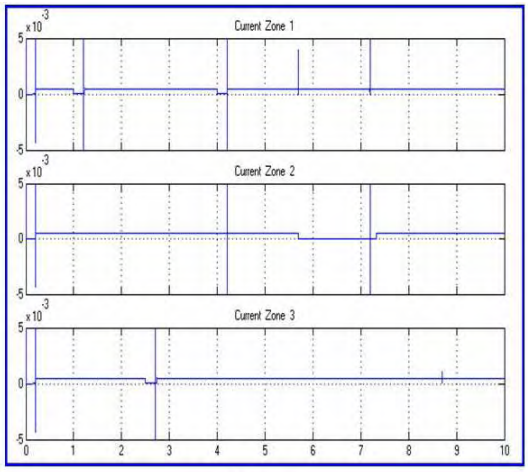


(c) Zone 3

(a) Time Diagram of Control Signals



(a) Voltage



(a) Current

(b) Zone Voltages and Currents.

Figure 5.20: BDB-DCZS Scenario II - Different Priority - Simulation Results.

- At t=0.3 seconds, a fault takes place in the SSCM of Zone2, connected to the Port Bus. Zone2 has the SSCM connected to the Starboard bus available, so the connection is changed and the load is fed using resources from the Starboard bus.
- At t=0.45 seconds, a fault takes place in the SSCM of Zone3, connected to the Port Bus. Zone2 has the SSCM connected to the Starboard bus available, so the connection is changed and the load is fed using resources from the Starboard bus.
- At t=0.6 seconds, a fault takes place in the SSCM of Zone2 connected to the Starboard Bus. The system only has two SSCMs available: the SSCM of Zone1 connected to the Port Bus and the SSCM of Zone3 connected to the Starboard Bus. Only two loads can be fed and they are selected based on the priority scheme.
- At t=0.8 seconds, a fault in the SSCM of Zone3 connected to the Port bus is cleared. Now the system can be fed by connection from the zones using the SSCM of Zone1 and Zone3 connected to the Port Bus.

5.4.1.1 Test with Default Priority Scheme

For the first simulation of this scenario, we used the default priority scheme. Figure 5.21 shows the Sniffer Agent GUI.

When the system starts, each connection between the zones in the Simulink™ model and their ServerAgents in the MAS is open, and each ServerAgent sends a SUBSCRIBE message to its ZoneAgent. During the time of the simulation the client socket in the Simulink™ Model sends information regarding the status of the SSCM to the ServerAgent.

When a fault in the SSCM of Zone1 connected to the Starboard Bus occurs at t=0.2 sec., ServerAgent1 sends a REQUEST message to ZoneAgent1 to inform the agent about the fault. The zone is not using this connection, so ZoneAgent1 saves this information

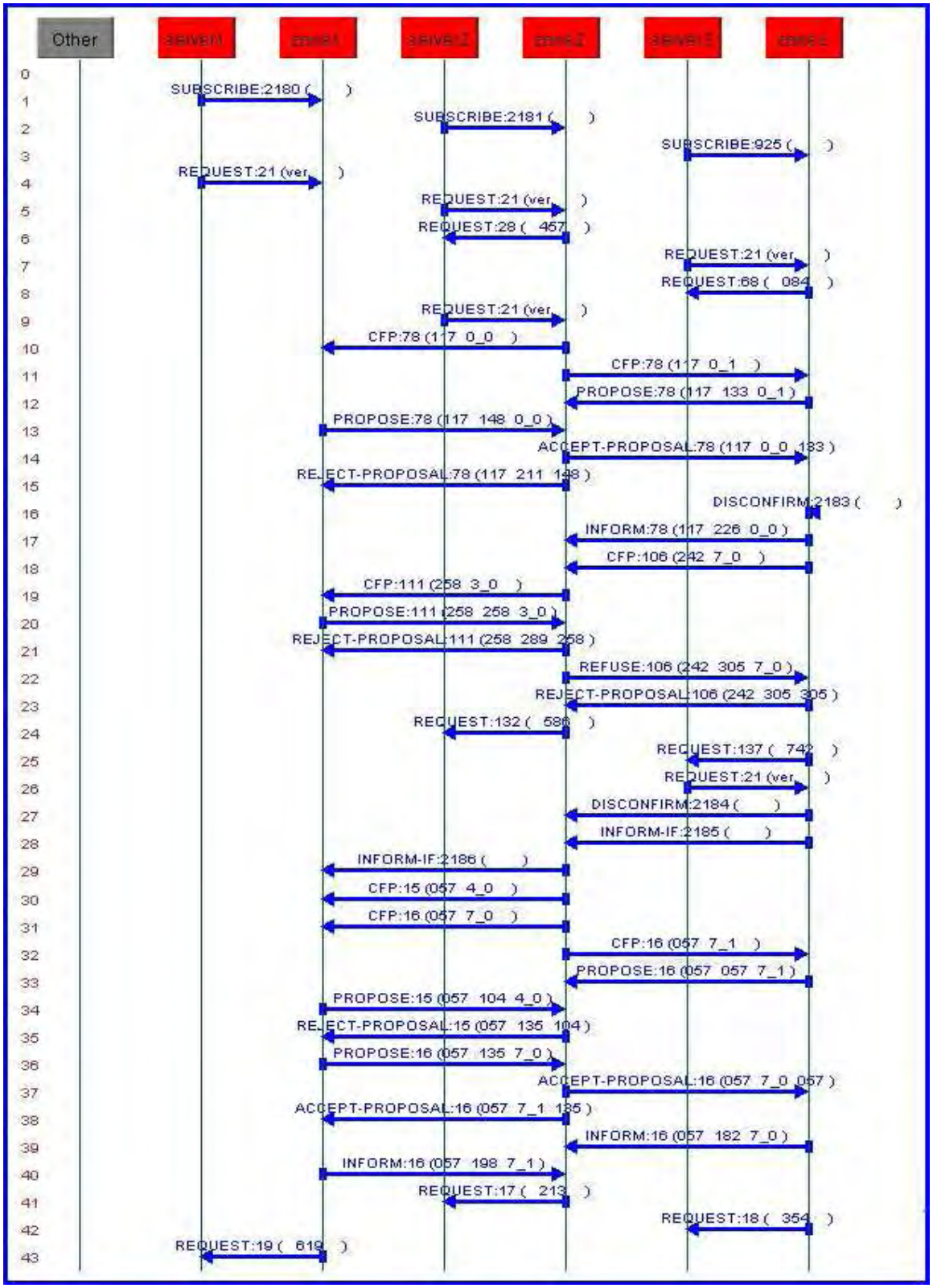


Figure 5.21: RDB-DCZS Scenario I - Default Priority: Sniffer Agent GUI - Negotiation Process.

but no action is taken at this point.

At t=0.3 sec. and t=0.45 sec, faults occur, in the SSCMs of Zone2 and Zone3 connected to the Port Bus, respectively. In both cases, the zones have the connection to the Starboard bus available, so the ZoneAgents make the decision to feed the load with the SSCM connected to the Starboard bus.

At t=0.6 seconds, a fault takes place in the SSCM of Zone2 connected to the Starboard Bus. ZoneAgent2 receives the notification from ServerAgent2 and starts a negotiation process. ZoneAgent2 sends CFP messages to ZoneAgent1 and ZoneAgent3. ZoneAgent1 does not have enough resources, and its priority is higher than the priority of Zone2; so, it creates a proposal for a partial amount of the power requested by ZoneAgent2. ZoneAgent3 creates a proposal based on disconnecting its load according to the priority scheme. ZoneAgent2 rejects ZoneAgent1's proposal and sends an ACCEPT_PROPOSAL message to ZoneAgent3. ZoneAgent3 disconnects its load. ZoneAgent3 tries to find resources to feed its load and it starts a negotiation process with ZoneAgent2 who redirects de CFP to ZoneAgent1. The negotiation process does not provide positive results because the system is reconfigured to feed loads with the highest priority, in this case, Zone1 and Zone2.

When a fault in the SSCM of Zone3 connected to the Port bus is cleared (at t=0.8 seconds), the three loads can be fed by the connection of the zones using the SSCM of Zone1 and Zone3 connected to the Port Bus. ZoneAgent3 connects its load and informs its neighbor ZoneAgent2 about the new resources. ZoneAgent2 was connected by the Starboard Bus of ZoneAgent3 so, to find a lowest cost option it starts a Negotiation Process with ZoneAgent1 and ZoneAgent3. Once the negotiation is complete the tie switches between the zones are closed and the system can feed the three loads.

After the reconfiguration actions are decided, in each step of the process, each

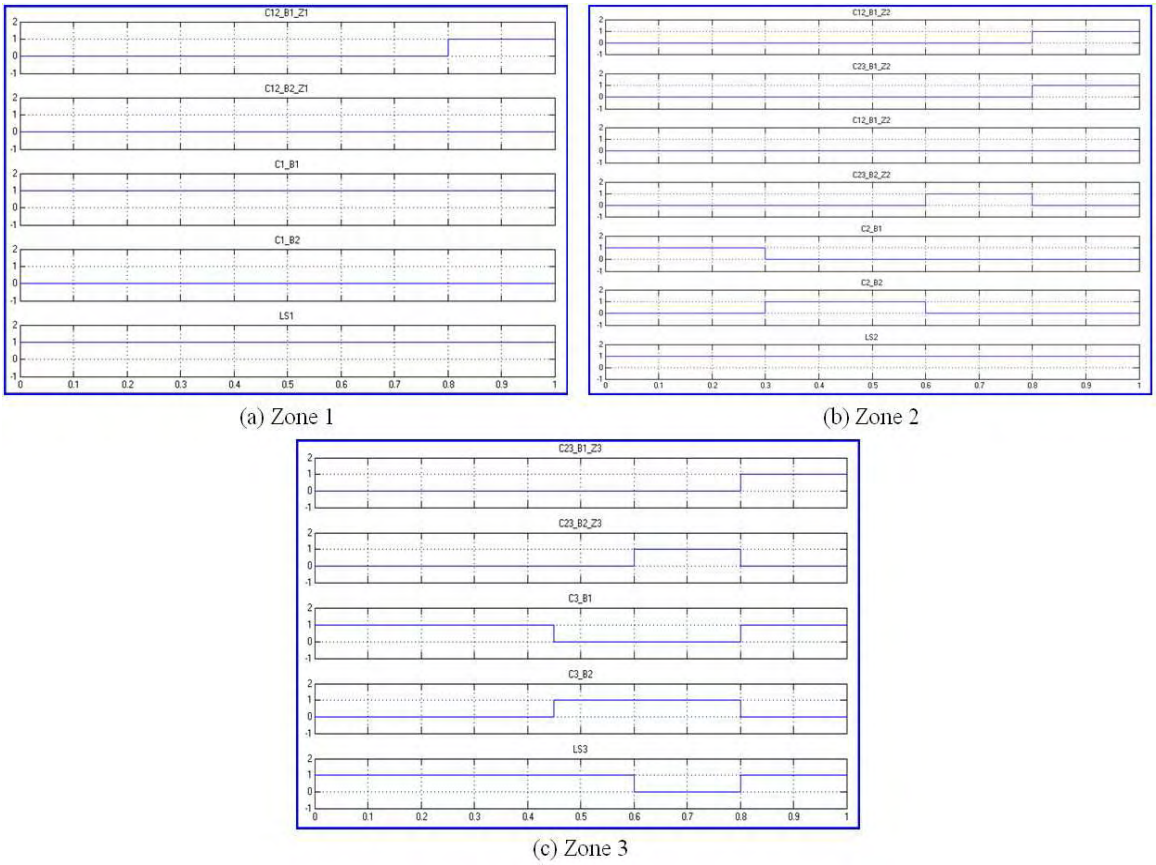


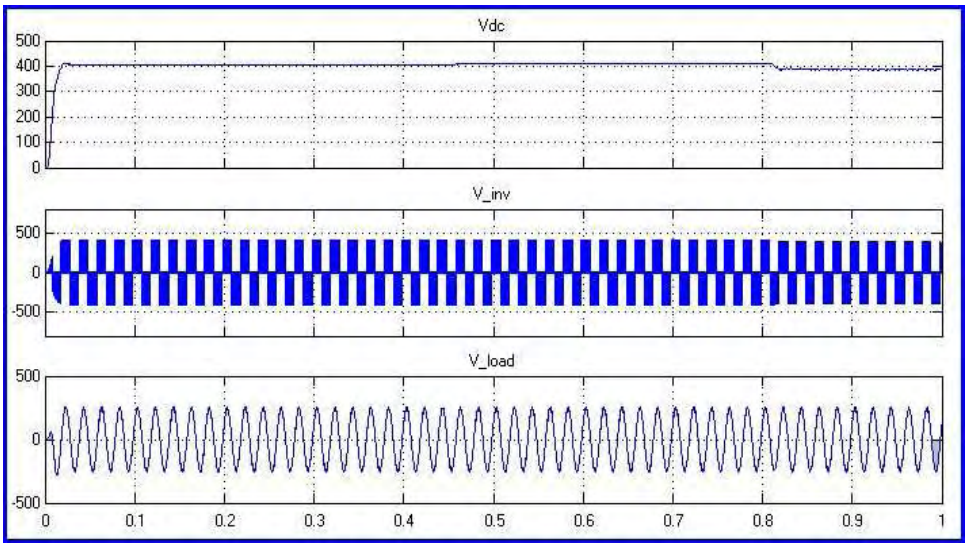
Figure 5.22: RDB-DCZS Scenario I - Default Priority - Time Diagram of Control Signals.

ZoneAgent establishes the control signals and sends them to its ServerAgent, who transmits them to the Simulink^{TM} model. Figure 5.22 shows the control signals for each zone; sent from ZoneAgents to ServerAgents. The Figure shows the expected behavior based on the described fault conditions. Figure 5.23 shows the output waveforms of each zone.

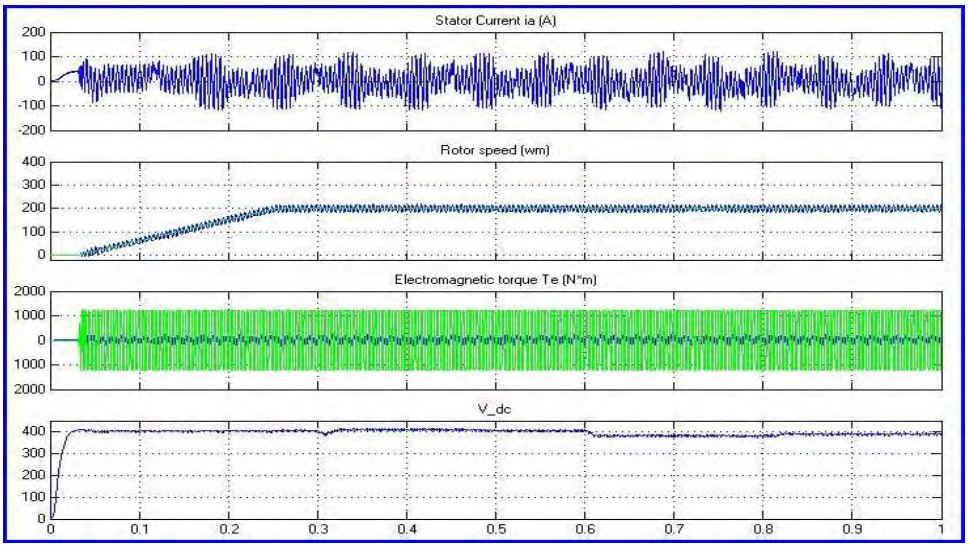
5.4.1.2 Test with Different Priority Scheme

In this part of the simulated scenario: Zone1 has the lowest priority 3, Zone2 has the highest priority 1, and Zone3 has priority 2. We keep the parameters of each scenario; we only change the priority scheme. Figure 5.24 shows the Sniffer Agent GUI.

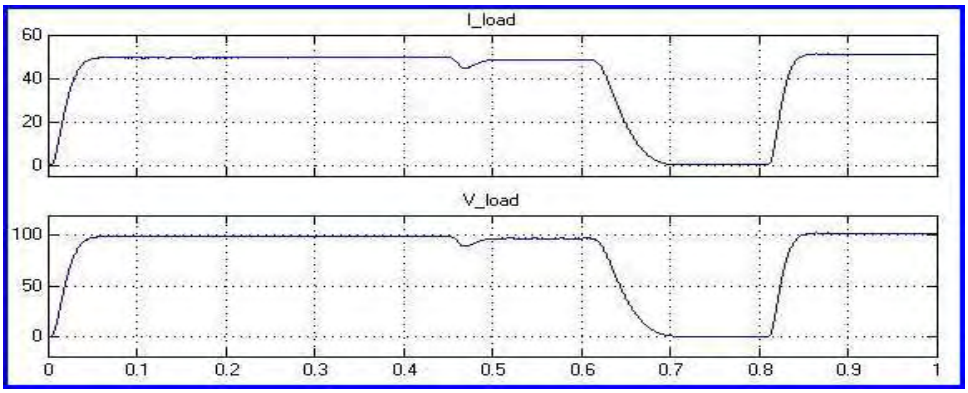
The first part of the process operates in a similar way to the one presented above.



(a) Zone1 Output waveforms.



(b) Zone2 Output waveforms.



(c) Zone3 Output waveforms.

Figure 5.23: RDB-DCZS Scenario I - Default Priority - Simulation Results.

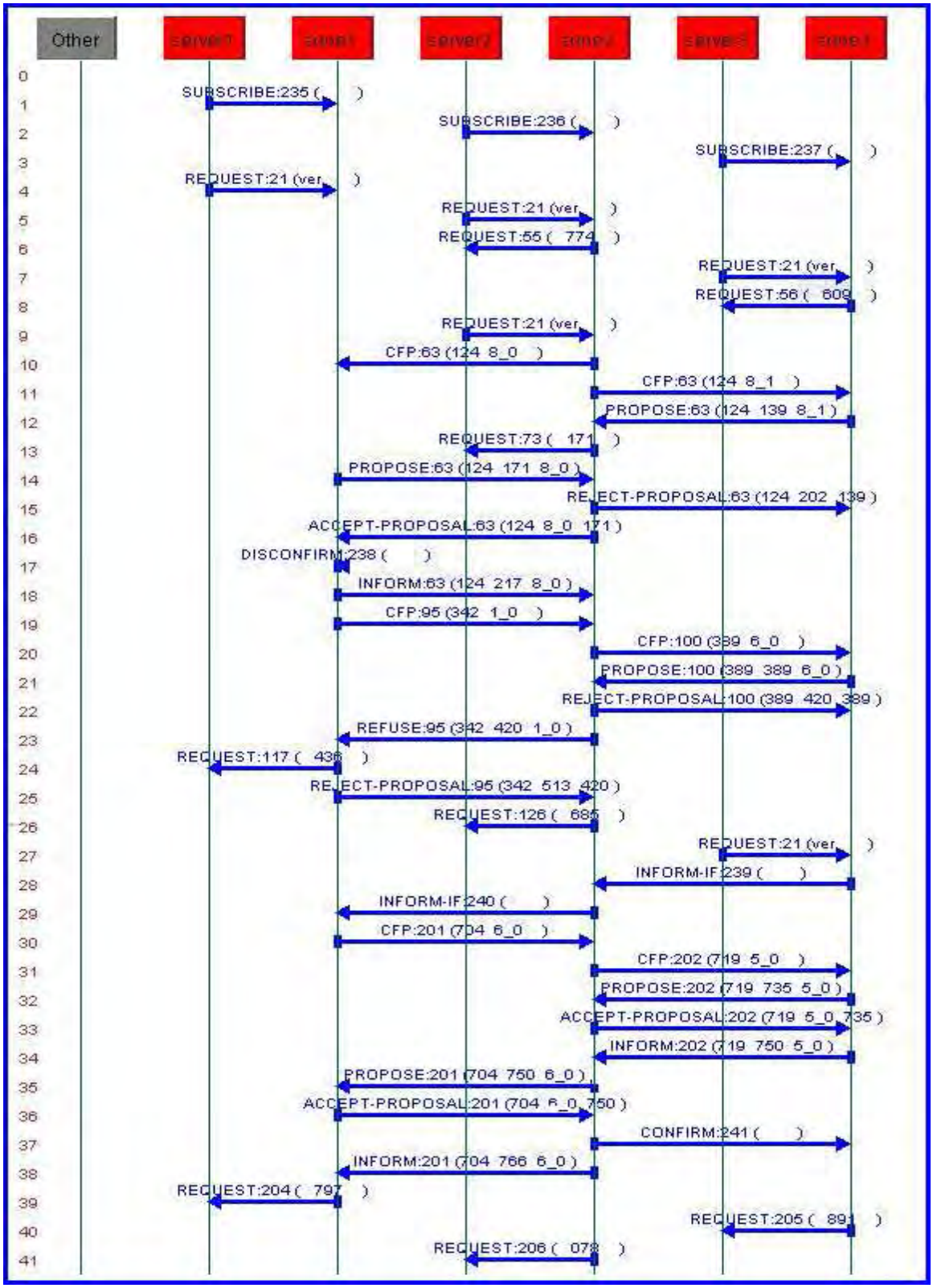


Figure 5.24: RDB-DCZS Scenario I - Different Priority: Sniffer Agent GUI - Negotiation Process.

At t=0.6 sec., when a fault in the SSCM of Zone2 connected to the Starboard Bus takes place, the system only can feed two loads. With the new priority scheme these loads are Zone2 and Zone3. ZoneAgent2 sends a CFP messages to ZoneAgent1 and ZoneAgent3. Both ZoneAgents send a proposal based on the disconnection of its load according to the priority scheme. ZoneAgent2 takes the resources of Zone1 because it has the lowest priority. ZoneAgent1's proposal is accepted and ZoneAgent3's proposal is rejected. ZoneAgent1 disconnects its load and tries to find resources but it cannot find a positive response. The load of Zone2 is fed by Zone1 and the load of Zone3 is fed by its own connection.

When a fault in the SSCM of Zone3 connected to the Port bus is cleared (at t=0.8 seconds), the three loads can be fed by the connection of the zones using the SSCM of Zone1 and Zone3 connected to the Port Bus. ZoneAgent3 informs its neighbor ZoneAgent2, who redirects the information to ZoneAgent1. ZoneAgent1 starts a negotiation process, this time with positive results. Following the negotiation, the tie switches between the zones are closed and the system can feed the three loads.

Figure 5.25 shows the control signals for each zone. Figure 5.26 shows the output waveforms of each zone.

5.4.2 RDB-DCZS - Scenario II

The general parameters for Scenario II are:

- Duration of Simulation: 1 second. The time of the simulation was fixed in order to facilitate visualization of the resulting figures.
- At the beginning of the simulation the system is in normal state (no faults).
- At t=0.2 seconds, a fault takes place in the SSCM of Zone2, connected to the Port Bus. Zone2 has the SSCM connected to the Starboard bus available, so the connection is changed and the load is fed using resources from the Starboard bus.

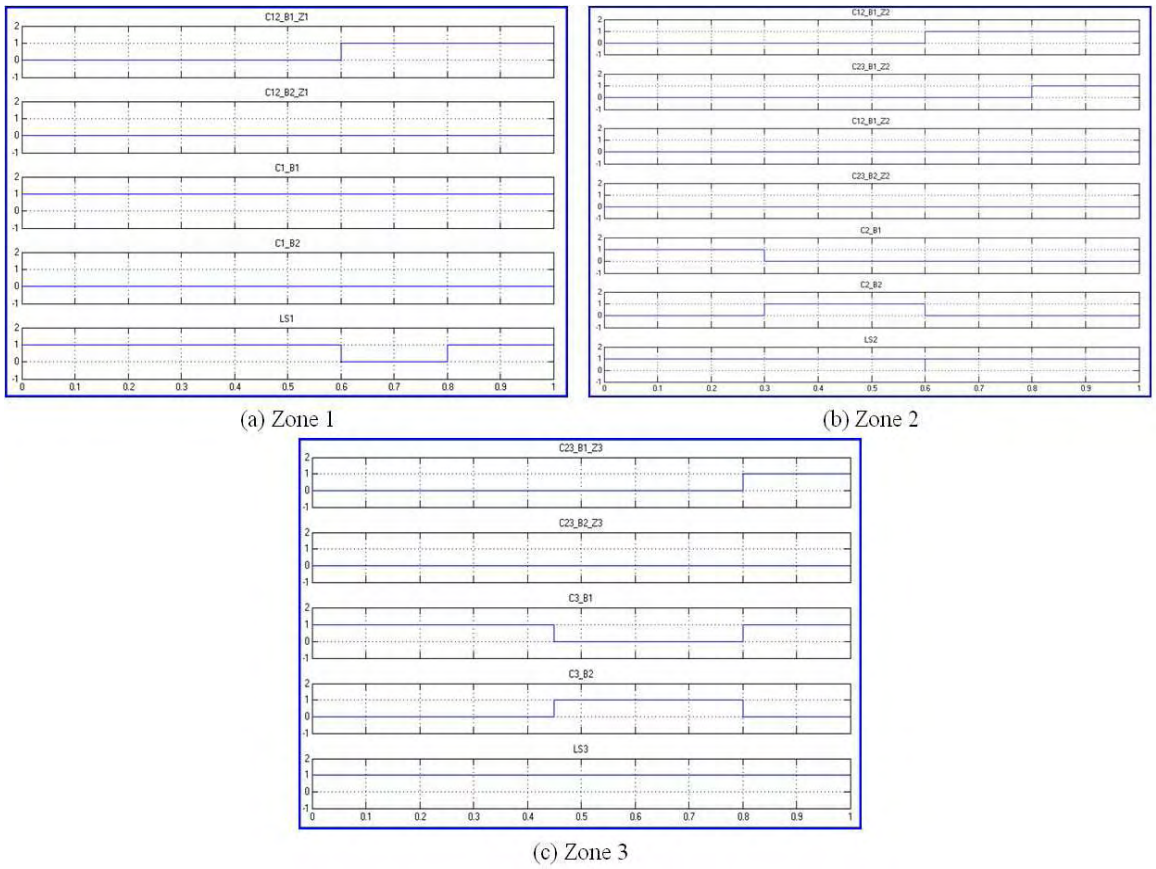
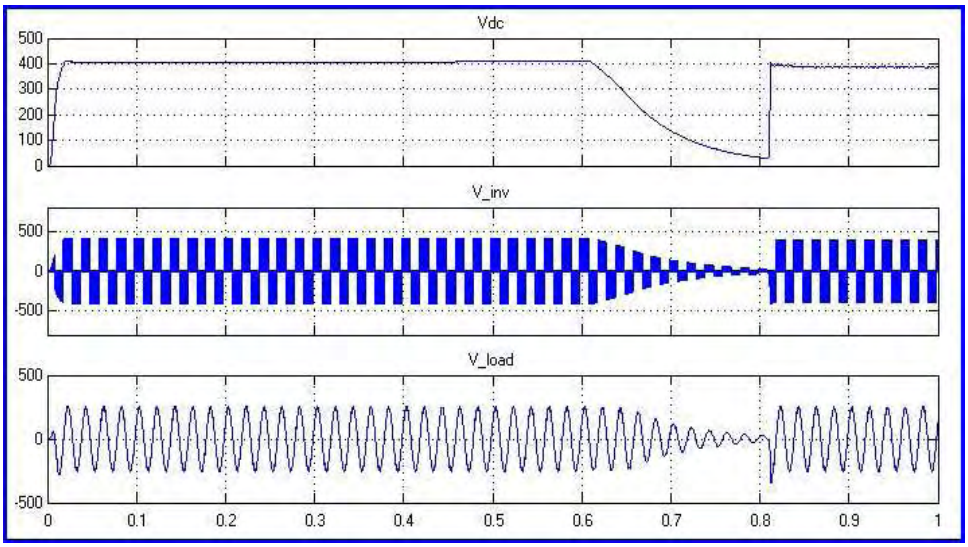
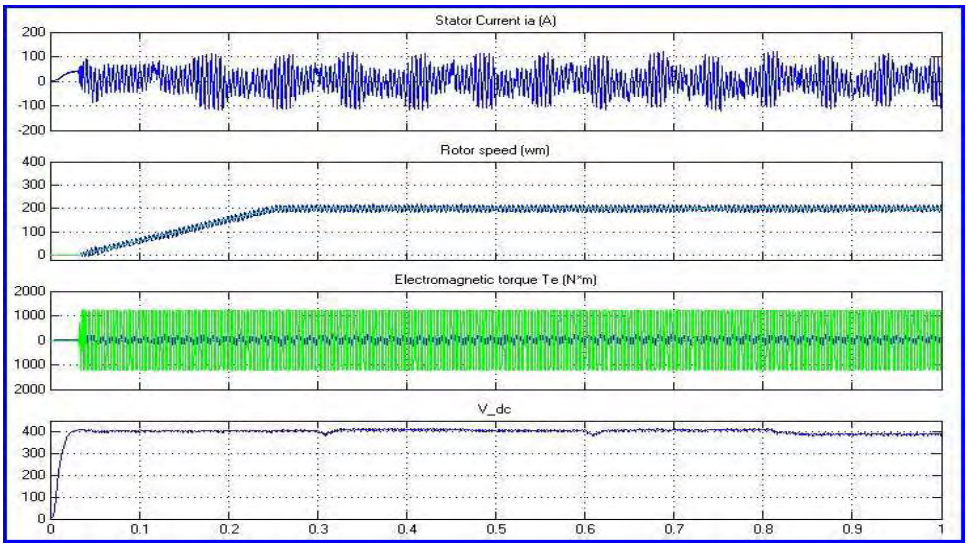


Figure 5.25: RDB-DCZS Scenario I - Different Priority - Time Diagram of Control Signals.

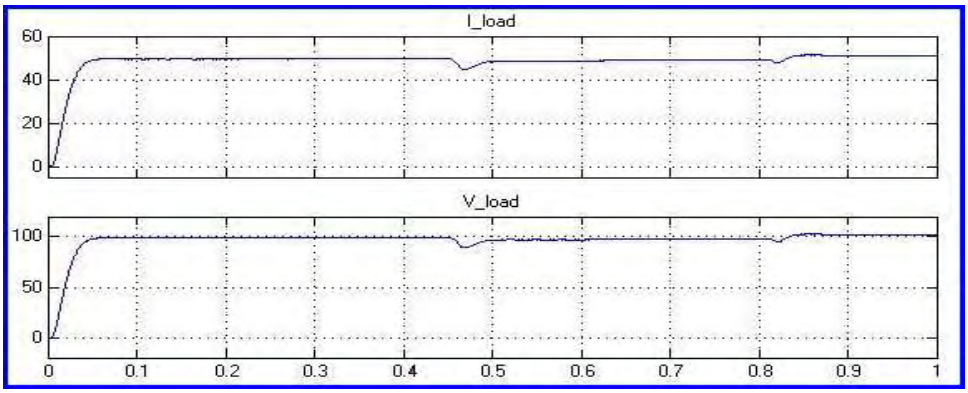
- At t=0.3 seconds, a fault takes place in the SSCM of Zone2, connected to the Starboard Bus. The tie switches between the zones are closed and the system can feed the three loads.
- At t=0.5 seconds, a fault takes place in the SSCM of Zone1, connected to the Port Bus. Zone1 has the SSCM connected to the Starboard bus available, so it moves its clients to this connection.
- At t=0.6 seconds, a fault takes place in the SSCM of Zone3 connected to the Starboard Bus. The system only has two SSCMs available: the SSCM of Zone1 connected to the Starboard Bus and the SSCM of Zone3 connected to the Port Bus. Only two loads can be fed and they are selected based on the priority scheme.



(a) Zone1 Output waveforms.



(b) Zone2 Output waveforms.



(c) Zone3 Output waveforms.

Figure 5.26: RDB-DCZS Scenario I - Different Priority - Simulation Results.

- At t=0.8 seconds, a fault takes place in the SSCM of Zone3 connected to the Port bus. The system only can feed one load and it is selected based on the priority scheme.

5.4.2.1 Test with Default Priority Scheme

For the first simulation of this scenario, we used the default priority scheme. Figures 5.27 and 5.28 show the Sniffer Agent GUI (Part 1 and Part 2).

When a fault in the SSCM of Zone2 connected to the Port Bus takes place (at t=0.2 sec.), the zone has the connection to the the Starboard bus available, so the ZoneAgent makes the decision to feed the load with the SSCM connected to the Starboard bus.

At t=0.3 seconds, a fault takes place in the SSCM of Zone2, connected to the Starboard Bus. Zone2 does not have connections available. ZoneAgent2 sends CFP to ZoneAgent1 and ZoneAgent3. Both ZoneAgents send proposals that are accepted. The tie switches between the zones are closed and the system can feed the three loads.

When a fault takes place in the SSCM of Zone1 connected to the Port Bus (at t=0.5 sec.), ZoneAgent1 must move its clients to connection with the Starboard Bus. One of the clients of ZoneAgent1 is ZoneAgent2, so ZoneAgent1 sends a REQUEST_WHEN message to request a change. ZoneAgent2 needs to coordinate with all its helpers. ZoneAgent2 sends a REQUEST_WHEN message to ZoneAgent3. The agents perform the change and the clients are now connected using the Starboard bus.

At t=0.6 seconds, a fault takes place in the SSCM of Zone3 connected to the Starboard Bus. Zone3 has the connection to the Port bus available and is helping Zone2 using the Starboard bus, so ZoneAgent3 sends a REQUEST_WHEN message to ZoneAgent2, who also asks ZoneAgent1 using a REQUEST_WHEN message. But, ZoneAgent1 cannot move the connection and ZoneAgent2 must ask for an alternative. ZoneAgent2 initiates a nego-

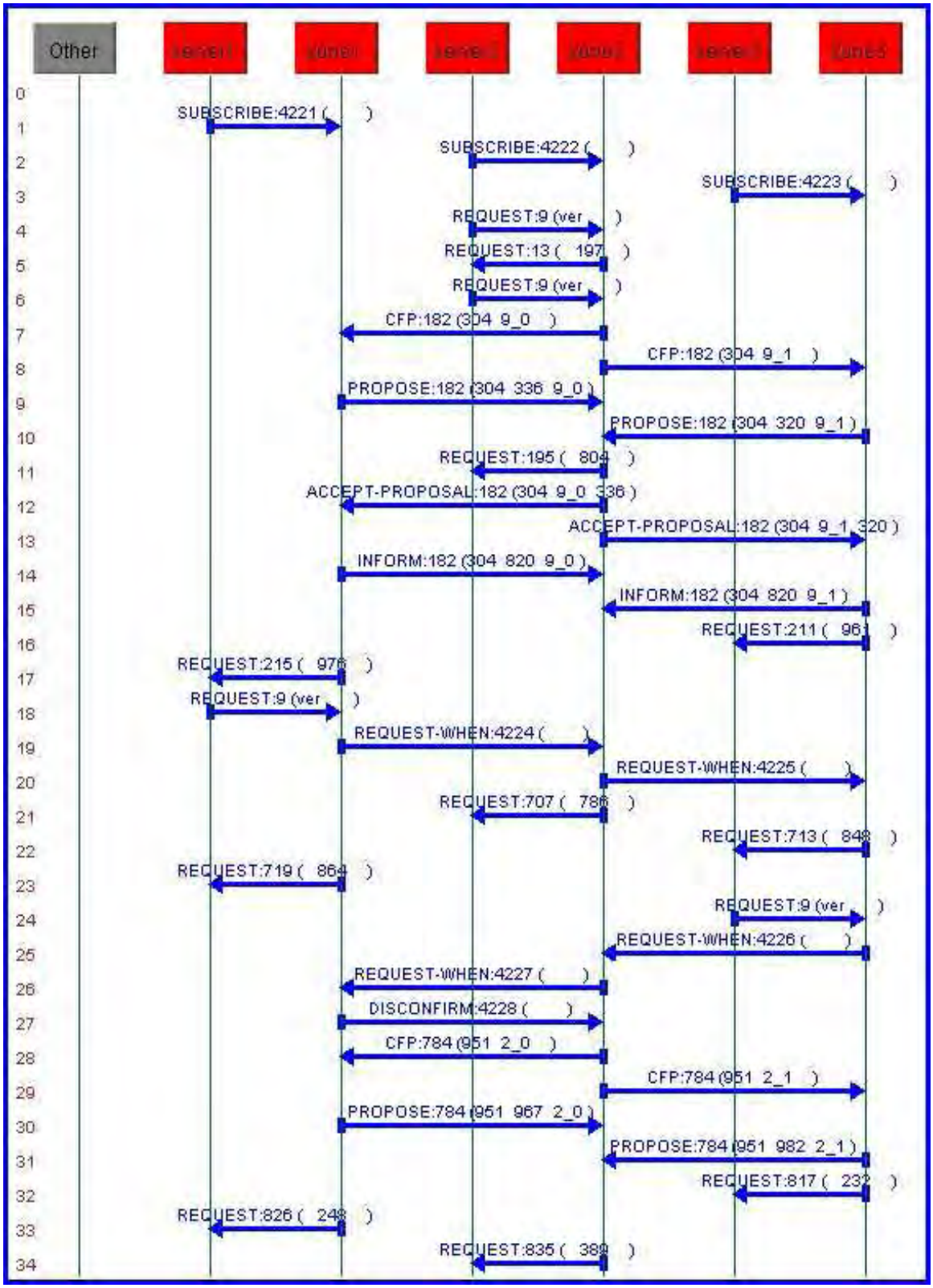


Figure 5.27: RDB-DCZS Scenario II - Default Priority: Sniffer Agent GUI - Negotiation Process (Part 1).

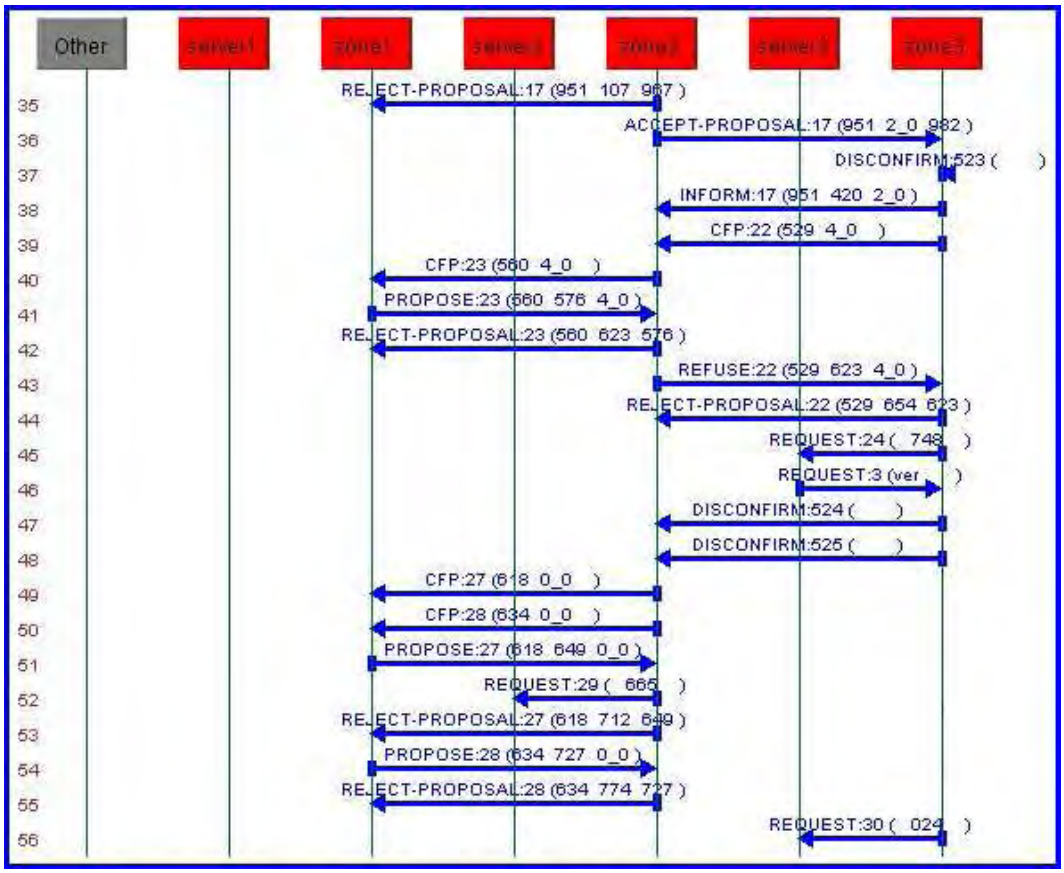


Figure 5.28: RDB-DCZS Scenario II - Default Priority: Sniffer Agent GUI - Negotiation Process (Part 2).

tiation process and based on the priority scheme, it accepts a proposal from ZoneAgent3 who disconnects its load to serve the load of Zone2.

At t=0.8 seconds, a fault takes place in the SSCM of Zone3 connected to the Port bus. ZoneAgent3 sends a DISCONFIRM message to ZoneAgent2 because it cannot help the zone. ZoneAgent2 starts a negotiation process but cannot find help because the SSCM of Zone3 is in fault condition and Zone1 is serving only its load because it has the highest priority.

Figure 5.29 shows the control signals for each zone. This Figure shows the expected behavior based on the described fault conditions. Figure 5.30 shows the output waveforms

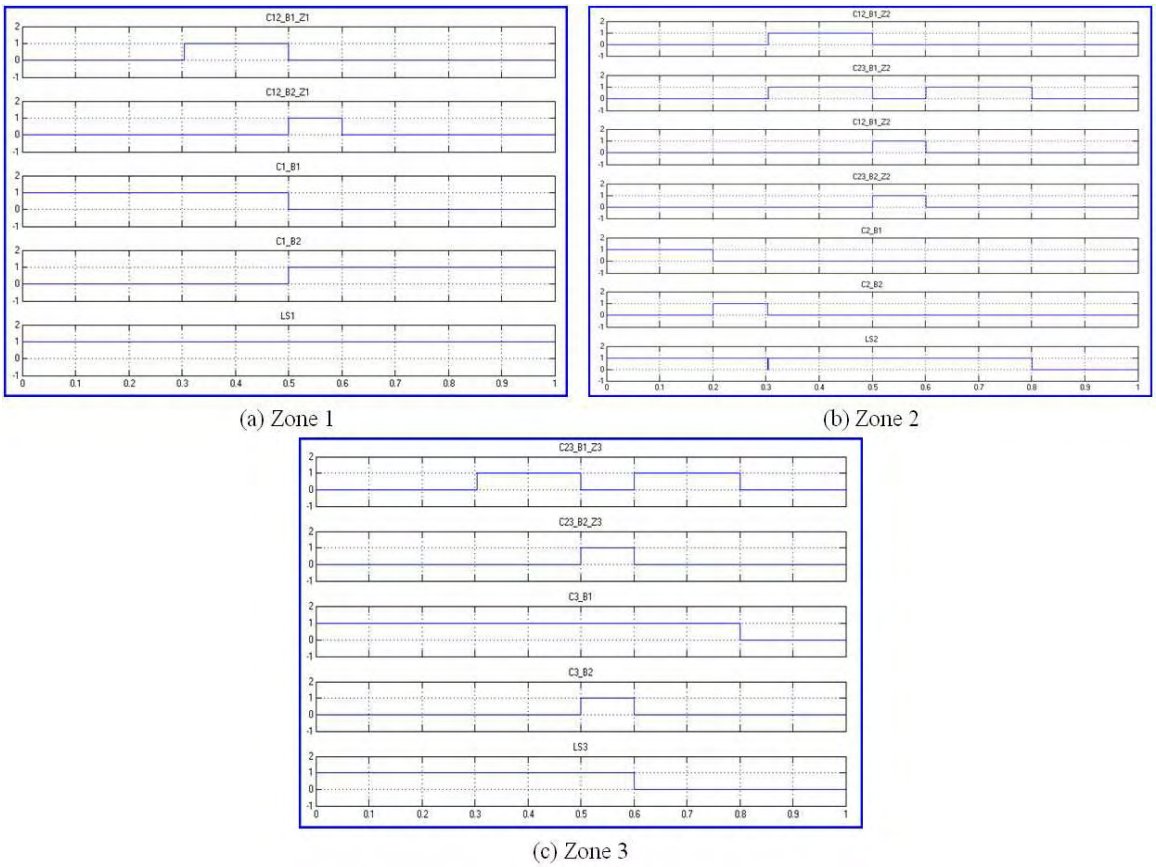


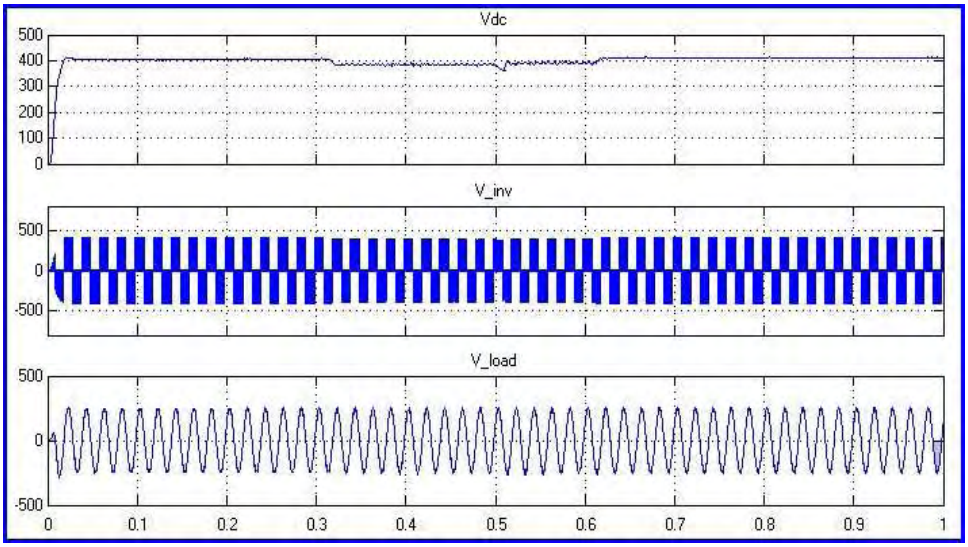
Figure 5.29: RDB-DCZS Scenario II - Default Priority - Time Diagram of Control Signals.

of each zone.

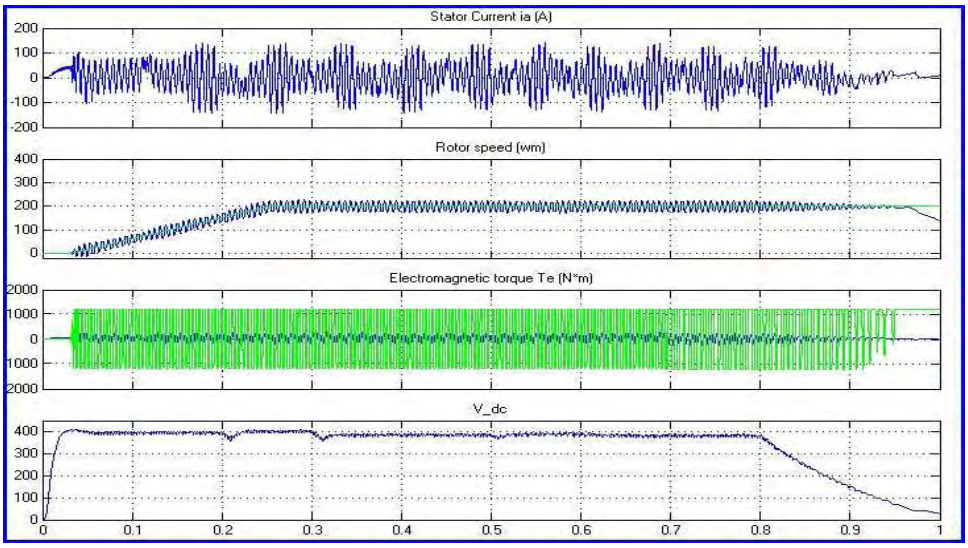
5.4.2.2 Test with Different Priority Scheme

In this simulation scenario: Zone1 has priority 2, Zone2 has the lowest priority 3, and Zone3 has the highest priority 1. We keep the parameters of each scenario; we only change the priority scheme. Figures 5.31 and 5.32 show the Sniffer Agent GUI (Part 1 and Part 2).

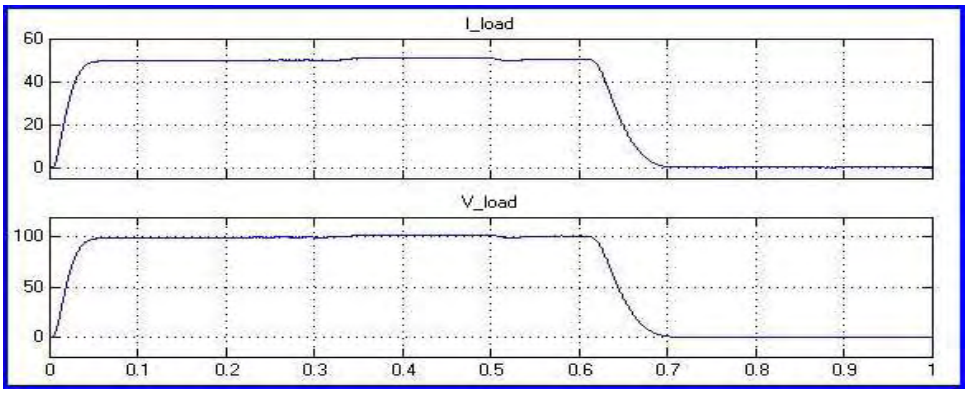
The first part of the process operates in the same way as to the one presented previously. When a fault in the SSCM of Zone3 connected to the Starboard Bus takes places, at t=0.6 seconds, Zone3 has the connection to the Port bus available and it is helping Zone2 us-



(a) Zone1 Output waveforms.



(b) Zone2 Output waveforms.



(c) Zone3 Output waveforms.

Figure 5.30: RDB-DCZS Scenario II - Default Priority - Simulation Results.

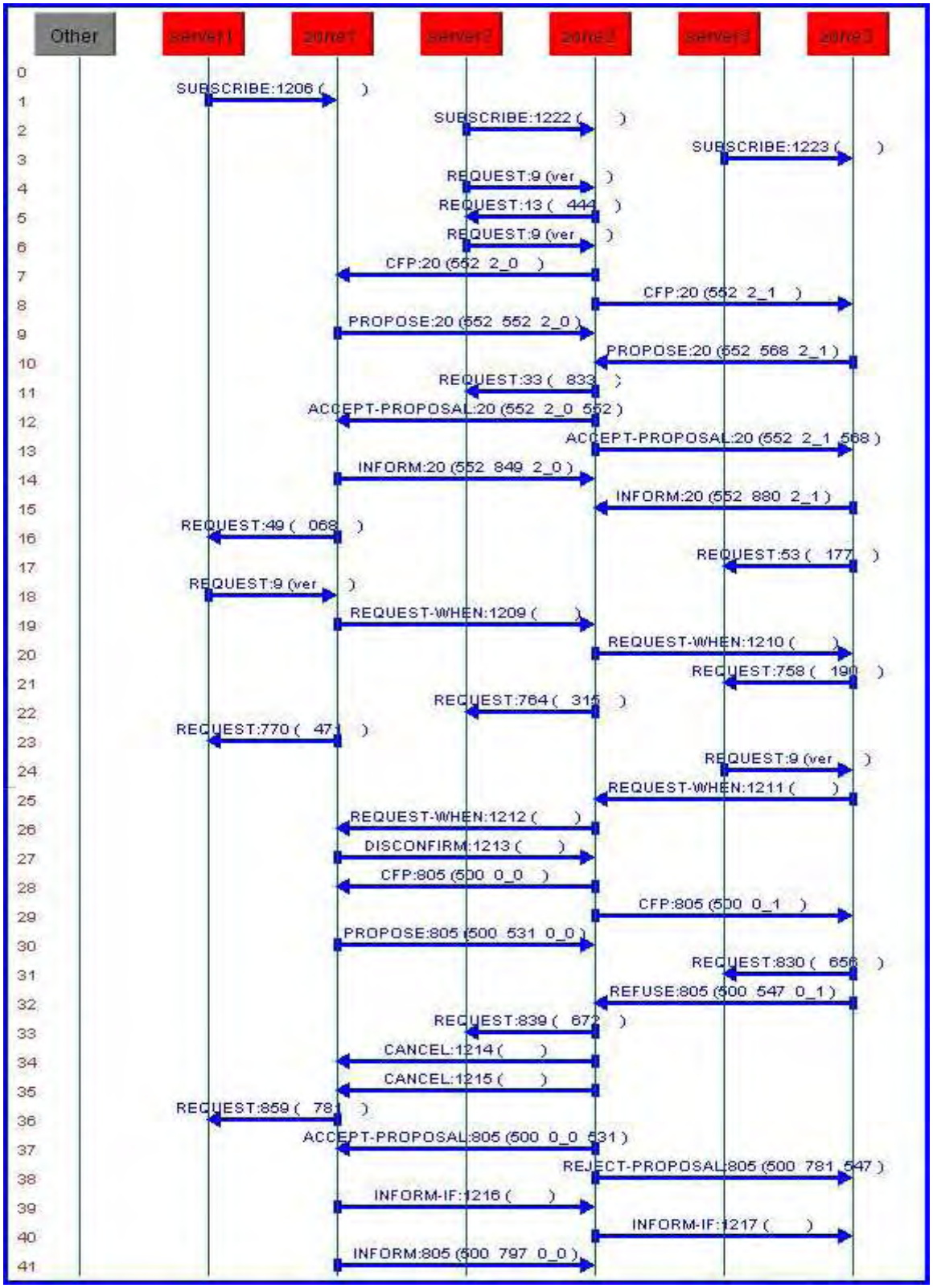


Figure 5.31: RDB-DCZS Scenario II - Different Priority: Sniffer Agent GUI - Negotiation Process (Part 1).

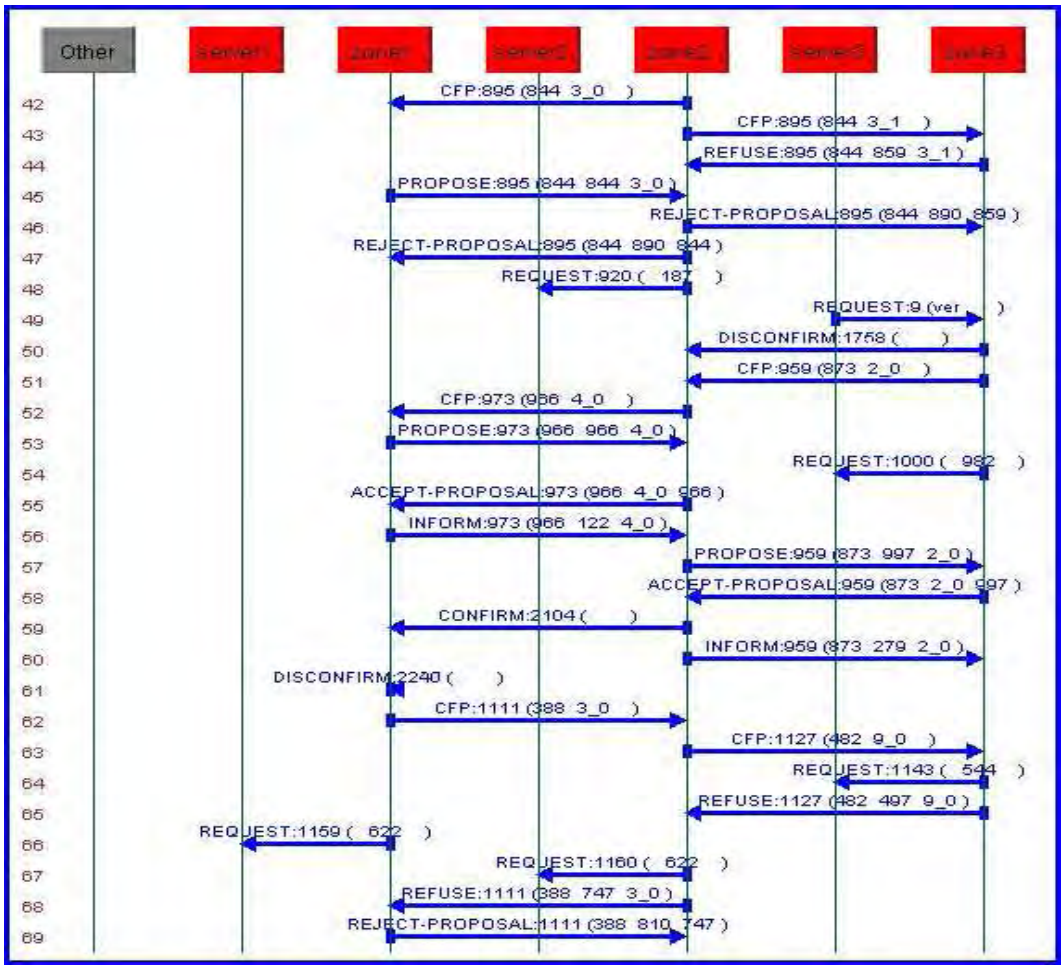
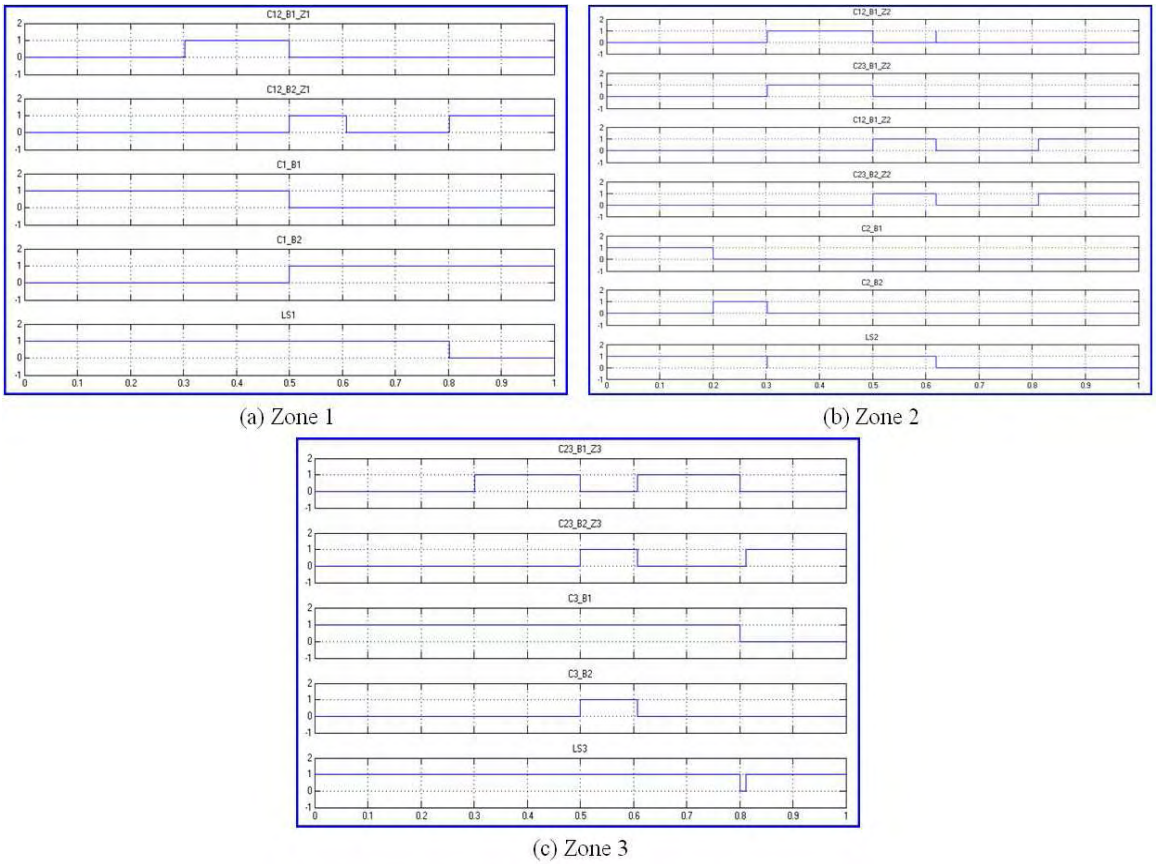


Figure 5.32: RDB-DCZS Scenario II - Different Priority: Sniffer Agent GUI - Negotiation Process (Part 2).

ing the Starboard bus, so ZoneAgent3 sends a REQUEST_WHEN message to ZoneAgent2, who also asks ZoneAgent1 using a REQUEST_WHEN message. But, ZoneAgent1 cannot move the connection and ZoneAgent2 must ask for an alternative. ZoneAgent2 initiates a negotiation process based on the priority scheme, but it does not receive proposals because Zone1 and Zone3 do not have enough resources and they have higher priority than Zone2.

At t=0.8 seconds, a fault takes place in the SSCM of Zone3 connected to the Port bus. ZoneAgent3 starts a negotiation process with ZoneAgent2. ZoneAgent2 does not have resources so it sends a CFP message to ZoneAgent1 to subcontract resources to Zone3.

Based on the priority scheme, ZoneAgent1 makes a proposal. ZoneAgent2 sends a proposal to ZoneAgent3 with the resources from ZoneAgent1. The proposal is accepted and the system is reconfigured to feed the load of Zone3 using the resources of Zone1. ZoneAgent1 tries to find resources to feed its load but is not successful, due the priority scheme.

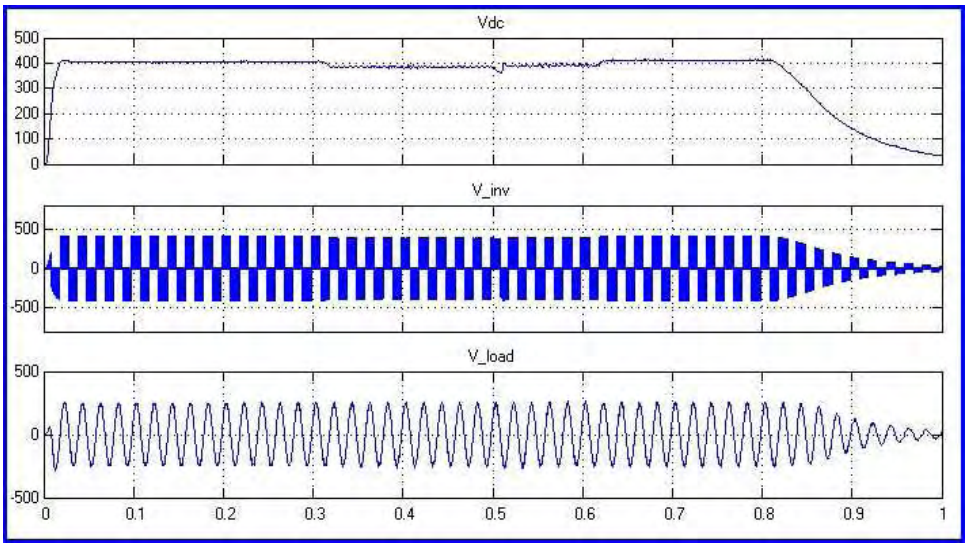


(a) Zone 1 (b) Zone 2

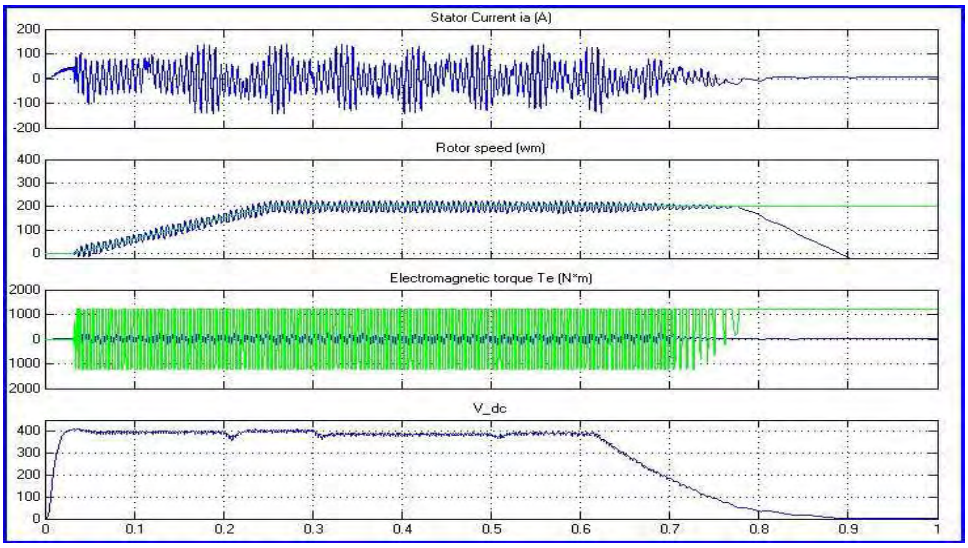
(c) Zone 3

Figure 5.33: RDB-DCZS Scenario II - Different Priority - Time Diagram of Control Signals.

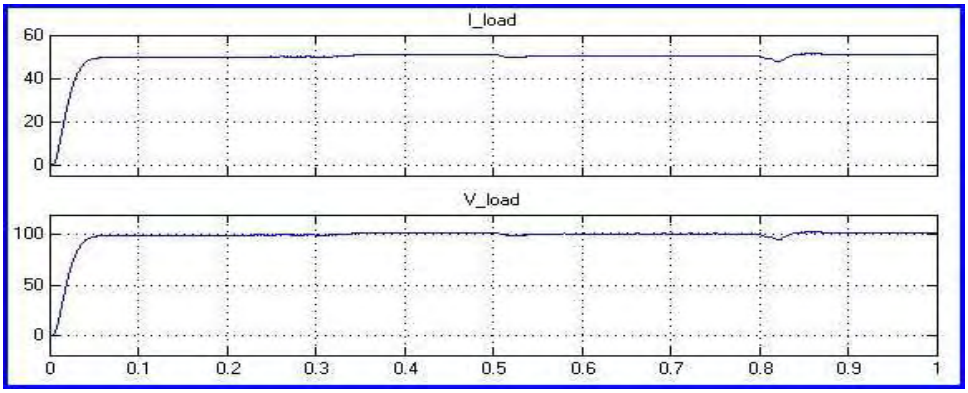
Figure 5.33 shows the control signals for each zone. Figure 5.34 shows the output waveforms of each zone.



(a) Zone1 Output waveforms.



(b) Zone2 Output waveforms.



(c) Zone3 Output waveforms.

Figure 5.34: RDB-DCZS Scenario II - Different Priority - Simulation Results.

5.5 Conclusions

In this Chapter, we presented two simulation scenarios for each test system to demonstrate the effectiveness of the proposed MAS-based EPDS. Each scenario was presented with the default priority scheme and with an alternative priority scheme. In every scenario the MAS was capable of reconfiguring the system in order to maximize the number of loads served with the highest priority. In addition, when there was more than one option to serve a load, the agents interacted to find the lowest cost option in each case.

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

We have presented the design of a FIPA-complaint MAS-based EPDS, using the concept of IPR. IPRs are distributed over the Zonal Electrical System with one IPR per zone. Each Zonal IPR is composed of two agents which interact in a MAS to reconfigure a zonal electric distribution system. The proposed MAS responds when a fault event occurs and through negotiation between agents reconfigures the system to maximize the number of loads served with highest priority. Also, the MAS has the capacity to respond when a fault is cleared, allowing the system to restore power to loads that were disconnected to supply power to higher priority loads. The restore resources are distributed based on the priority scheme. When there is more than one alternative to feed a load, the MAS connects its load to the lowest cost option.

We implemented a MAS prototype using the JADE platform and JAVA programming language. To test the prototype, we integrated it with three simulation test electrical systems. The test systems were implemented using Simulink^{TM} - SimPowerSystems toolbox and are based on zonal architectures. In order to understand the reconfiguration process, we used a simple electrical system based on a single bus zonal architecture for our first experiments. Then we moved to a simple electrical system based on a dual bus architec-

ture, where the issue of redundancy was added to the problem and solution. The third test system is based on the Simulink™ implementation of the DC Zonal Electric Distribution System (DCZEDS) provided by the Office of Naval Research (ONR) to researchers in the NSF/ONR EPNES program [20]. The last test system provided a realistic vision of the MAS operation.

The MAS and the simulation test system were implemented using different technologies, because it allows us to use the appropriate language for each part of the system. Therefore, it was necessary to communicate Simulink™ with JADE platform. Due to the limitations of the Simulink™ platform, we used sockets to implement the communication middleware. For each zone, one agent in the MAS acted as a Server (ServerAgent), and one Embedded Matlab™ Function in the simulation electrical system acted as a client. Through the experiments we found some synchronization problems between Simulink™ and Java due to time back stepping in the integration routines used in Simulink™. We proposed a practical solution to these synchronization problems.

Through simulation, we evaluated the performance of the MAS-based EPDS to assess that the MAS-based EPDS reconfigures an electrical system to maximize the number of loads served with highest priority and to minimize the cost of resources when there is more than one alternative to feed a load. We designed some test scenarios and each was run with the default priority scheme and with an alternative scheme, to show the independence between the MAS operation and the priority scheme.

Our simulation results show the feasibility of implementing a self-reconfigurable EPDS using a MAS approach, since it provides autonomous capabilities to the system and it allows agents to negotiate without full knowledge of the system.

6.2 Future Work

Future efforts can be dedicated towards:

- Improving the communication interface between Simulink™ and Java in the Simulation Model. Including capabilities to send/receive more data types, and to pre-process the data in order to send only relevant information to the Multi-Agent System.
- Establishing a fault detection strategy to integrate with the reconfiguration logic.
- Introducing Power Quality issues as part of the decision making process.
- Improving response time by pre-computing a set of reconfiguration cases of common contingencies.
- Testing the MAS using a larger version of a zonal system.
- Integrating the Zonal IPR with previous developments in the restoration and reconfiguration area of the IPR research project.
- Implementing the verification capabilities of the CommunicationAgent. This implementation must be integrated with the propose MAS prototype.
- Analyzing the costs sensitivity to normalize the factors and to improve the proposed cost function used during the decision process.
- Evaluating the use of Distribute Database techniques to facilitate the synchronization between agents and into the process of each agent.
- Improving the reconfiguration process using control techniques, such as, Model Predictive Control (MPC). MPC allows the prediction of the future behavior of the system for a specific time horizon, based on the current state of the system, the expected disturbances and the planned control signal [52]. MPC is aimed to find the control signals that optimize the objective function, and it is possible to evaluate, in advance, their performance based on the objective function.

The objective function in MPC could be optimized using Optimal Control [54]. This is the standard method for solving dynamic optimization problems, if they are based on continuous time. When Optimal Control is used, it is possible to find a control law for a system in order to achieve an optimality criterion.

The agents could use MPC to control their zones by making the predictions about the evolution of the connections over a horizon time. Based on this new scheme, it is fundamental to settle the information to communicate, the moments when it must be shared and to make the decisions of the control actions [53]. The communication between agents needs to be improved and the information in the negotiation process must be evaluated.

- The MAS proposed in this research work was designed and applied to EPDS with zonal architectures based on single and dual bus distribution. In [17] a variety of zonal architectures is presented and they represent the state of the art of zonal ship design; therefore, the future work could be focused in the design of a more complex MAS that could be used in various zonal architectures. In addition, it is important to evaluate and extend the work to other EDPS architectures, such as ring, radial and so on.

BIBLIOGRAPHY

[1] Butler, K.L.; Sarma, N.D.R.; Whitcomb, C.; DoCarmo H. and Zhang, H. "Shipboard Systems Deploy Automated Protection", *IEEE Computer Application in Power*, Vol. 11, Issue 2, pp. 31-36, April 1998

[2] Irizarry-Rivera A.; Rodrǵuez M.; Vélez-Reyes M.; Cedeńo J.R.; Vélez B.; ONeill-Carrillo E.O. and Ramírez A. "Intelligent Power Routers for Distributed Coordination in Electric Energy Processing Networks". EPNES Workshop, October 23-24 2003. Orlando, Florida.

[3] Vergara-Laurens, I.J. "A Decentralized Negotiation Framework for Restoring Electrical Energy Delivery Networks with Intelligent Power Routers - IPRs". M.S. Thesis. University of Puerto Rico, Mayagüez, P.R., 127 pp. 2005.

[4] Jennings, N.R. and Bussman, S. "Agent-Based Control Systems: Why are They Suited to Engineering Complex Systems?", *IEEE Control Systems Magazine*, Vol. 23, No. 3, pp. 61-73, June 2003.

[5] Wikipedia, the free encyclopedia. "Software agent". (2006) [Online]. Available: http://en.wikipedia.org/wiki/Software_agent/

[6] The Foundation for Intelligent Physical Agents. (2006) [Online]. Available: http://www.fipa.org.

[7] Telecom Italia Lab. "JADE (Java Agent DEvelopment Framework)". (2006) [Online]. Available: http://jade.tilab.com/

[8] Gómez-Gualdrón, J.G. and Vélez-Reyes, M. "Simulating a Multi-Agent Based Self-Reconfigurable Electric Power Distribution System". In *Proceedings of the* 10^{th} *IEEE Workshop on Computers in Power Electronics (COMPEL06)*. July 2006.

[9] Gómez-Gualdrón, J.G. and Vélez-Reyes, M. "A Multi-Agent approach for a Self-Reconfigurable electronic Power Distribution System". In *Proceedings of Intelligent Computing: Theory and Applications IV, SPIE*. Volume 6229. April, 2006.

[10] Wooldridge, M. *An Introduction to Multi-Agent Systems*. John Wiley and Sons. England, 2002.

[11] Weiss, G. (ed.), *Multi-Agent Systems. A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, Cambridge, MA, 1999.

[12] Farooq Ahmad, H. "Multi-agent Systems: Overview of a New Paradimg for Distributed Systems". In *Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02)*. 2002.

[13] Foundation for Intelligent Physical Agents. "FIPA ACL Message Structure Specification." (2006) [Online]. Available: http://www.fipa.org/specs/fipa00061/.

[14] Foundation for Intelligent Physical Agents. "FIPA Interaction Protocols Specification." (2006) [Online]. Available: http://www.fipa.org/repository/ips.php3.

[15] Jennings, N.R. and Wooldridge, M.J. "Applications Of Intelligent Agents" in *Agent Technology Foundations, Applications, and Markets*. Springer-Verlag, Heidelberg, Germany (1998).

[16] Chapman, S.J. *Electric Machinery and Power System Fundamentals*. McGraw-Hill, 2001. 103

[17] Doerry, N, USN Capt. "Zonal Ship Design". *Naval Engineers Journal*. Volume 118, Number 1, Winter 2006, pp. 39-53(15). Publisher: American Society of Naval Engineers

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

[19] Zivi, E.L. "Integrated Shipboard Power and Automation Control Challenge Problem". In *Proceedings of 2002 Power Engineering Society Summer Meeting* Vol. 1, pp. 325-330, July 2002

[20] ONR / NSF EPNES Control Challenge Problem Website. (2006) [Online]. Available: http://www.usna.edu/EPNES.

[21] Estremera, A. and Velez-Reyes, M. "Design of a Reconfigurable DC Zonal Distribution System using an Event Driven Controller". In *Proceedings of 2005 Annual NSF Site Visit, Center for Power Electronic Systems*. April, 2005.

[22] Sudhoff, S.D.; Pekarek, S.; Kuhn, B.; Glover, S.; Sauer, J.; Delisle, D. "Naval combat survivability testbeds for investigation of issues in shipboard power electronics based power and propulsion systems". In *Proceedings of the 2002 Power Engineering Society Summer Meeting* Vol. 1 , 21-25 July 2002.

[23] Estremera, A. "Design of a Reconfiguration Controller for a DC Zonal Distribution System". M.S. Thesis. University of Puerto Rico, Mayagüez, P.R., 81 pp. 2005.

[24] Davey, K. and Hebner, R. "Reconfiguration of Shipboard Power Systems". *IASME Transactions*. Vol.1, pp. 241-246, April 2004.

[25] Srivastava, S.K. and Butler-Puny, K.L. "A Probability Based Predictive Reconfiguration Method for Shipboard Power Systems". In *Proceedings of* 8^{th} *International Conference on Probabilistic Methods Applied to Power Systems*. September, 2004.

[26] Jin X.; Zhao, J.; Sun, Y.; Li, K. and Zhang, B. "Distribution Network Reconfiguration for Load Balancing Using Binary Particle Swarm Optimization". In *Proceedings of 2004 International Conference on Power System Technology (POWERCON'2004)* November, 2004.

[27] Wikipedia, the free encyclopedia. "Genetic algorithm". (2006) [Online]. Available: http://en.wikipedia.org/wiki/Genetic_algorithm

[28] Mendoza, J.; Lpez, R.; Morales, D.; Lpez, E.; Dessante, P. and Moraga, R. "Minimal Loss Reconfiguration Using Genetic Algorithms With Restricted Population and Addressed Operators: Real Application". *IEEE Transactions on Power Systems*. Vol. 21, No. 2, May 2006.

[29] Salazar, H.; Gallego, R. and Romero, R. "Artificial Neural Networks and Clustering Techniques Applied in the Reconfiguration of Distribution Systems". *IEEE transactions on Power Delivery*. Vol. 21, No. 3, July 2006.

[30] Qiu-Xuan, W.; Guang-Yi, C. and Yan-Qiong, F. "Research on metamorphic algorithm of modular self-reconfigurable robots based cellular automata". In *Proceedings of the* 4^{th} *International Conference on Machine Learning and Cybernetics*. August, 2005.

[31] Feliciano-Bonilla, C. "A Time-domain simulatin framework of an IPR-based Shipboard Integrated Power system". M.S. Thesis. University of Puerto Rico, Mayagüez, P.R., 123 pp. 2006.

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

[33] Thorp, J.S.; Wang, X.R.; Hopkinson, K.M.; Coury, D.; Giovanini, R., "Agent Technology Applied to the Protection of Power Systems". *Autonomous Systems and Intelligent Agents in Power System Control and Operation*. Editor Christian Rehtanz. Baden, Schweiz: Springer-Verlag. 2003.

[34] Hopkinson, K.M.; Birman, K.P.; Giovanini, R.; Coury, D.V.; Wang, X.; Thorp, J.S., "EPOCHS: Integrated Commercial Off-The-Shelf Software for Agent-based Electric Power and Communication Simulation". In *Proceedings of 2003 Winter Simulation Conference*. December, 2003.

[35] Sun, L. and Cartes D. "A Multi-Agent System for Reconfiguration of Shipboard Power Systems". *IASME Transactions*. Issue 3, Volume 1, July 2004.

[36] G. Morejon, L. Sun, D. Cartes. "CORBA Implementation for Interfacing Software Agents with Virtual Test Bed". In *Proceedings of ASNE Reconfiguration and Survivability Symposium 2005*. February, 2005.

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

[38] Solanki, J.M. and Schulz, N.N. "Using Intelligent Multi-Agent Systems for Shipboard Power Systems Reconfiguration". In *Proceedings of the* 13^{th} *International Conference on Intelligent Systems Application to Power Systems (ISAP)*. November, 2005.

[39] Maturana, F., Staron, R. Discenzo, F., Scheidt, D., Pekala, M., Bracy, J. and Zink, M. "An Interoperable Agent-based Control System for Survivable Shipboard Automation". In *Proceedings of ASNE Reconfiguration and Survivability Symposium 2005*. February, 2005.

[40] Khushalani, S., and Schulz, N.N. "Restoration optimization with distributed generation considering islanding" In *Proceedings of the 2005 Power Engineering Society General Meeting*. Vol. 3, June, 2005.

[41] Smith R. G.: "The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver". *IEEE Transactions on Computers*, 1980.

[42] Foundation for Intelligent Physical Agents. "FIPA Contract Net Interaction Protocol Specification." (2006) [Online]. Available: http://www.fipa.org/specs/fipa00029.

[43] The MathWorks. "Simulink^{TM} - Simulation and Model-Based Design". (2006)[Online]. Available: http://www.mathworks.com/products/simulink/

[44] The MathWorks. "MATLAB - Simulink Documentation". (2006) [Online]. Available: http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/

[45] The MathWorks. "SimPowerSystems 4.3, Model and simulate electrical power systems". (2006) [Online]. Available: http://www.mathworks.com/products/simpower/

[46] Bellifemine, F.; Poggi, A. and Rimassa, G. "JADE: a FIPA2000 compliant agent development environment". In *Proceedings of the* 5^{th} *International Conference on Autonomous agents* p.216-217, May 2001.

[47] Bellifemine, F.; Caire, G.; Poggi, A. and Rimassa, G. "JADE A White Paper". *Telecom Italia EXP magazine*. Vol 3, No 3 September 2003.

[48] The MathWorks. "MATLAB Documentation, External Interfaces". (2006) [Online]. Available: http://www.mathworks.com/access/helpdesk/help/ techdoc/ matlab_external/

[49] Sun Microsystems, Inc. "The Java™ Tutorial". (2006) [Online]. Available: http://java.sun.com/docs/books/tutorial/

[50] E. Rusty Harold. *Java Network Programming*. 2^{nd} Edition. O'reilly 2000.

[51] MATLAB Central File Exchange. "TCP/UDP/IP Toolbox 2.0.5." (2006) [Online]. Available: http://www.mathworks.com/matlabcentral/ fileexchange/loadFile.do ?objectId=345&objectType=file

[52] Hegyi, A. "Model Predictive Control for Integrating Traffic Control Measures". Ph.D. Thesis. The Netherlands TRAIL Research School. 232 p. 2004.

[53] Negenborn, R.R.; De Schutter, B., and Hellendoorn, H. "Multi-agent model predictive control of transportation networks". In *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control (ICNSC 2006)*, Ft. Lauderdale, Florida, pp. 296-301, Apr. 2006.

[54] Wikipedia, the free encyclopedia. "Optimal control". (2006) [Online]. Available: http://en.wikipedia.org/wiki/Optimal_control

APPENDICES

APPENDIX A

Ship Service Converter Modules and Loads of Realistic Dual Bus DC Zonal System Model

As it was mentioned previously, the Realistic Dual Bus DC Zonal System Model (RDB-DCZS) is composed by two power supplies (PS), each one connected to a 500-V bus, one on the Starboard side and one on the Port side. Also, it contains six Ship Service Converter Modules (SSCM) connected to their load using an OR-ing diode. The load of Zone 1 is composed by one Ship Service Inverter Module (SSIM) and its associated Load Bank (LB). Zone 2 has a Motor Control (MC) module as load and Zone 3 has a Constant Power Load (CLP).

Next, we will present the description of the SSCM and the loads of each zone [19, 20, 22]. Also, we will introduce some details about the Simulink^{TM} - SimPowerSystems toolbox implementation.

A.1 Ship Service Converters Modules

The Ship Service Converter Module (SSCM) accepts 500 V DC, that may vary, and regulates the output voltage to 400 V DC for load currents up to 20 A. The rated

output power is 8 kW. Figure A.1 shows the circuit diagram of the Ship Service Converter Module (SSCM) [18, 19, 20]. Figure A.2 presents the SSCM circuit parameters.

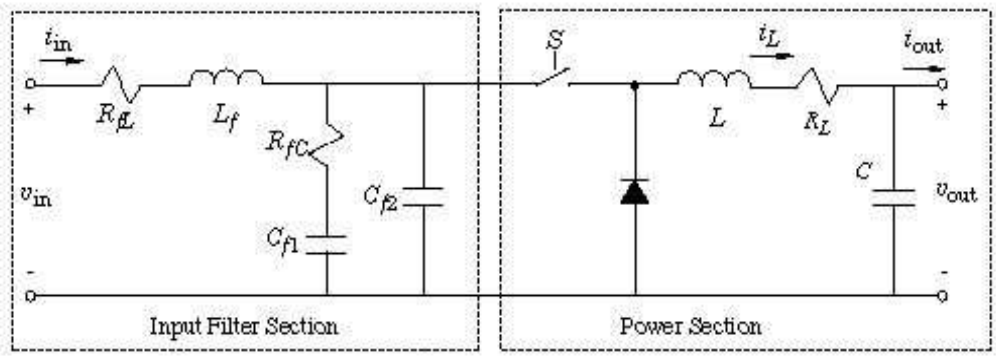


Figure A.1: Circuit diagram of the SSCM [19, 20, 22].

Parameter	Symbol	Value
Switching frequency	f_{sw}	20.0 kH
Rated dc input voltage	V_{in}	500.0 V
Rated dc output voltage	V_{out}	400.0 V
Rated output power	P_{out}	8.0 kW
Input filter inductor	L_f	357.0 μH
Input inductor resistance	R_{fL}	0.2 Ω
Input filter capacitor	C_{f1}	500.0 μF
Input filter series resistor	R_{fC}	1.0 Ω
Input filter additional capacitor	C_{f2}	45.0 μF
Buck converter dc inductor	L	3.0 mH
Resistance of the dc inductor	R_L	0.5 Ω
Buck converter output capacitor	C	500.0 μF

Figure A.2: SSCM circuit parameters [19, 20, 22].

In the SimPowerSystems Model, a Universal Bridge with a PWM Generator was used; it controls the duty cycle and work like a buck converter. Using one low-pass input filter and a output filter we have a DC-DC converter that convert from 500V to 400V. Figure A.3 presents the implementation of the SSCM in SimpowerSystems toolbox.

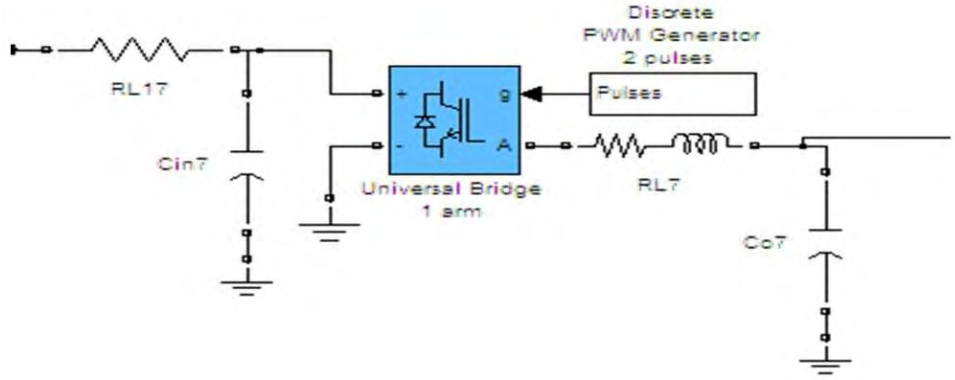


Figure A.3: The SSCM in the SimPowerSystems toolbox.

A.2 Load of Zone 1: SSIM and Load Bank

The load of Zone 1 is composed by one Ship Service Inverter Module (SSIM) and its associated Load Bank (LB). Figure A.4 shows a circuit diagram of the load of zone 1.

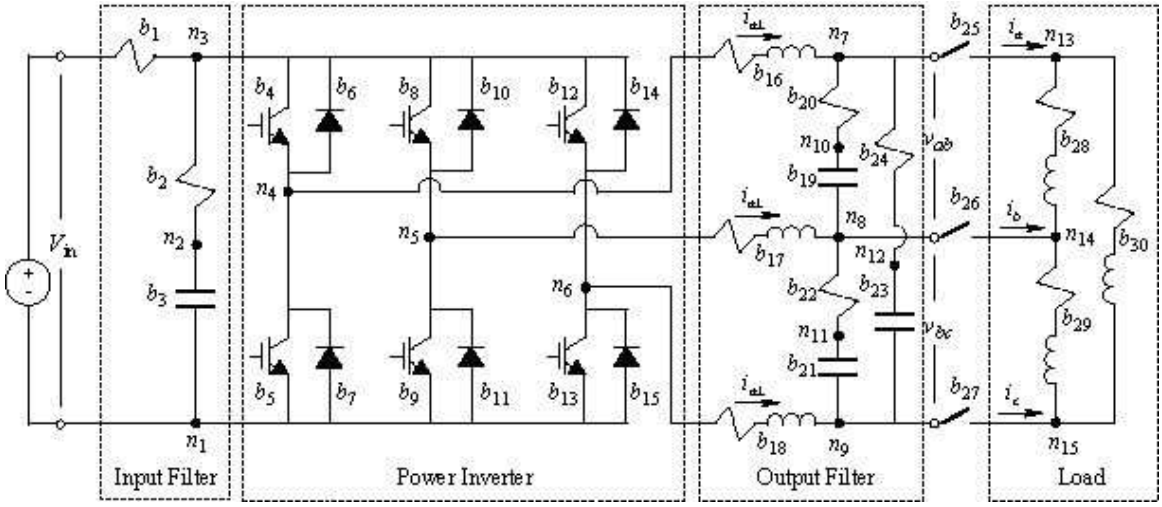


Figure A.4: Circuit diagram of the SSIM [19, 20, 22].

The Ship Service Inverter Module (SSIM) is a DC to AC Inverter that accepts a DC voltage with values between 380-440 V DC, and it provides balanced three-phase ac voltages of a specified frequency and magnitude for loads up to 10 kW. In this case, we use

400DCV to 250rmsACV [19, 20, 22]. Figure A.5 presents the circuit parameters.

Parameter	Symbol	Value	Branch(s)
Rated dc input voltage	V_{in}	380-440 V	1
Input filter, series resistance	R_s	10.0 mΩ	1
Input filter, capacitor series resistance	R_{cin}	127.0 mΩ	2
Input filter, capacitor	C_{in}	590.0 μF	3
Output filter, ac inductor	L_{ac}	550.0 μH	16, 17, 18
Output filter, ac inductor series resistance	R_{Lac}	38.0 mΩ	16, 17, 18
Output filter, ac capacitor	C_{ac}	50.0 μF	19, 21, 23
Output filter, ac capacitor series resistance	R_{Cac}	8.0 mΩ	20, 22, 24
Load, series inductance	L_{ld}	4.0 μH	28, 29, 30
Load resistance	R_{ld}	15.9 Ω	28, 29, 30

Figure A.5: SSIM circuit parameters [19, 20, 22].

In the SimPowerSystems Model, a PWM ISBT Inverter with a Discrete PWM Generator and a close loop system were used. This controls the duty cycle and permits a constant voltage output of 250 rms. Figure A.6 shows the SSIM in the SimPowerSystems toolbox.

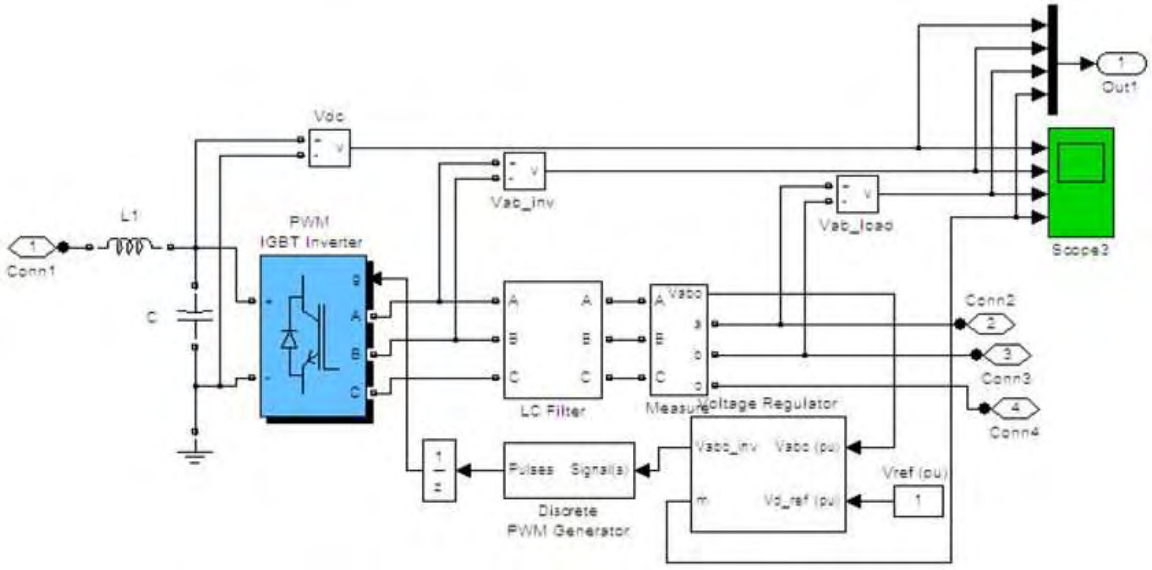


Figure A.6: The SSIM in the SimPowerSystems toolbox.

The load bank was implemented in SimPowerSystems toolbox, using a three phase Delta connection bank. Figure A.5 presents the parameters for the Load bank; they are the same parameters that the SSIM uses. Figure A.7 shows the circuit for the Load Bank.

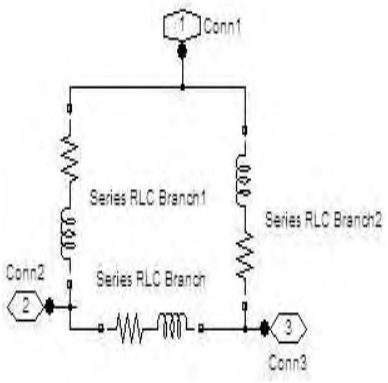


Figure A.7: The Load Bank in the SimPowerSystems toolbox.

A.3 Load of Zone 2: The motor controller

The motor controller is a SSIM with a three phase induction motor in the output. The MC inverter accepts a DC voltage that may vary between 300-420 V DC. The output of the inverter is connected to an induction motor. It is assumed that the motor model receives voltages as inputs and returns the corresponding stator currents [19, 20, 22]. Figure A.8 shows the Motor Controller Circuit, Figure A.9 presents the Motor Controller Circuit Parameters and Figure A.10 presents the Induction Motor Parameters.

In the SimPowerSystems Model, a Motor Controller was used. It has a Three-phase inverter with a Braking Chopper and a Speed Controller in a close loop system. This maintains a constant speed of 200 (rad/s) that can be change to a desired speed. Figure A.11 shows the Motor Controller in the SimPowerSystems Toolbox.

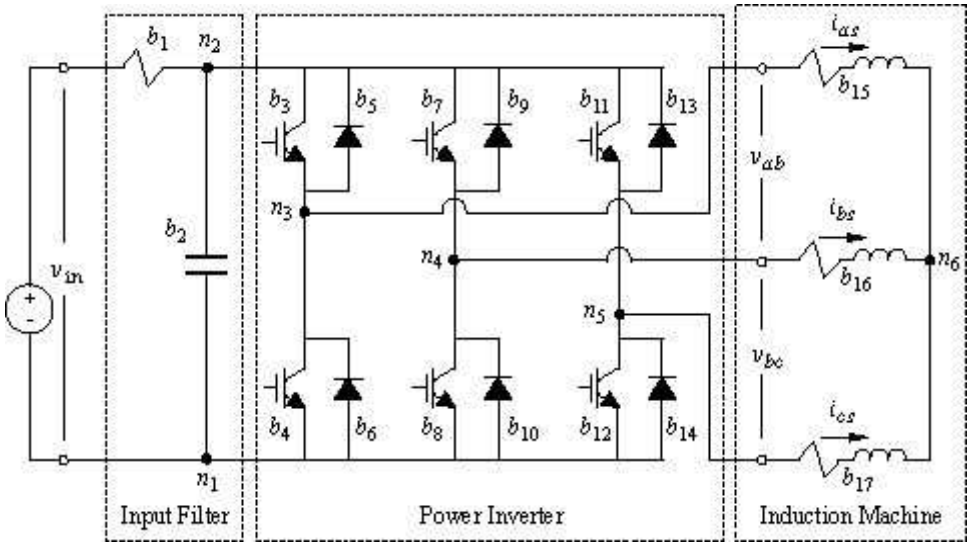


Figure A.8: Motor Controller Circuit [19, 20, 22].

Parameter	Symbol	Value	Branch(s)
Rated dc input voltage	V	300-400 V	1
Input filter, series resistance	r	10.0 mΩ	1
Input filter, capacitor	C	1000.0 μF	2

Figure A.9: Motor Controller circuit parameters [19, 20, 22].

A.4 Load of Zone 3: Constant Power Load

The Constant Power Load (CPL) accepts DC voltage that may vary between 120-600 V DC, and regulates the output voltage to 100 V DC. The converter is loaded with a 2-Ohm resistor, which at the given output voltage, dissipates 5 kW of power. Since the output voltage is tightly regulated to a constant level, the dissipated power is also constant. A CPL to convert from 400Vdc to 100Vdc is used [19, 20, 22]. Figure A.12 shows the circuit for the Constant Power Load. Figure A.13 presents the Constant Power Load circuit parameters.

Figure A.14 shows the SimPowerSystems toolbox implementation of Constant

Parameter	Symbol	Value
Rated dc input voltage	V	230 V (line-to-line, rms)
Base frequency	ω_{base}	377 r/s
Base flux	Ψ_{base}	185 V
Base torque	T_{base}	19.8 N-m
Number of poles	P	4
Mechanical inertia	J	0.089 $kg \cdot m^2$
Stator winding resistance	r_s	0.315 Ω
Stator winding leakage reactance	X_{ls}	0.546 Ω
Magnetizing inductance	X_m	18.92 Ω
Rotor winding resistance	r_r	0.591 Ω
Rotor winding leakage reactance	X_{lr}	0.546 Ω

Figure A.10: Induction Motor Parameters [19, 20, 22].

Power Load. A DC-DC buck converter with a resistance of 2 ohms was used to implemented.

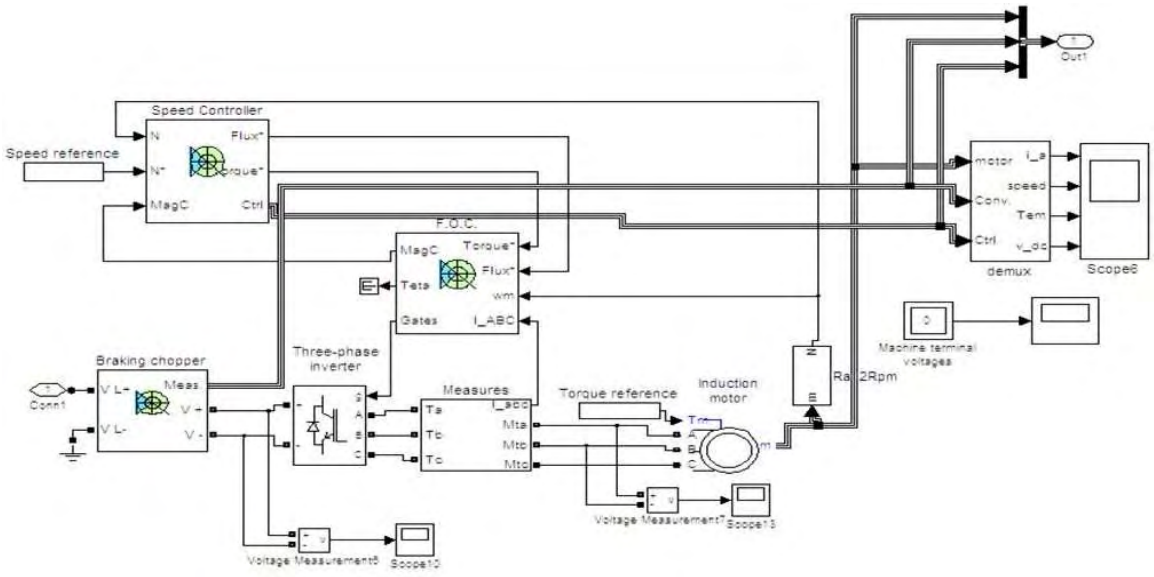


Figure A.11: Motor Controller in the SimPowerSystems Toolbox.

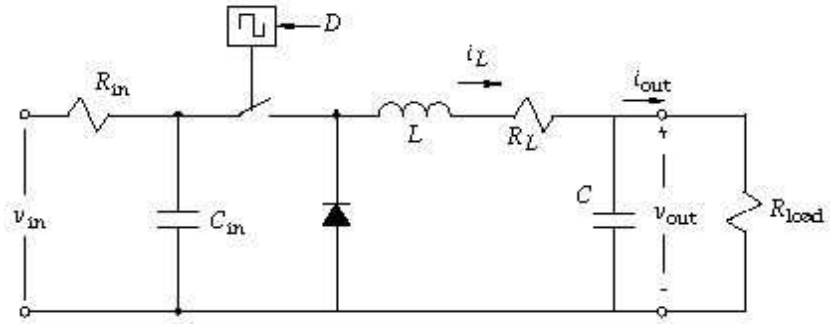


Figure A.12: Constant Power Load Circuit [19, 20, 22].

Parameter	Symbol	Value
Rated dc input voltage	v	120-600.0 V
Rated dc output voltage	v	100.0 V
Rated output power	P	5.0 kW
Input filter resistance	R	1.0 Ω
Input filter capacitor	C	470.0 μF
Buck converter dc inductor	L	2.0 mH
Resistance of the dc inductor	R_{dc}	0.01 Ω
Buck converter output capacitor	C	470.0 μF
Nominal load resistance	R	2.0 Ω

Figure A.13: Constant Power Load circuit parameters [19, 20, 22].

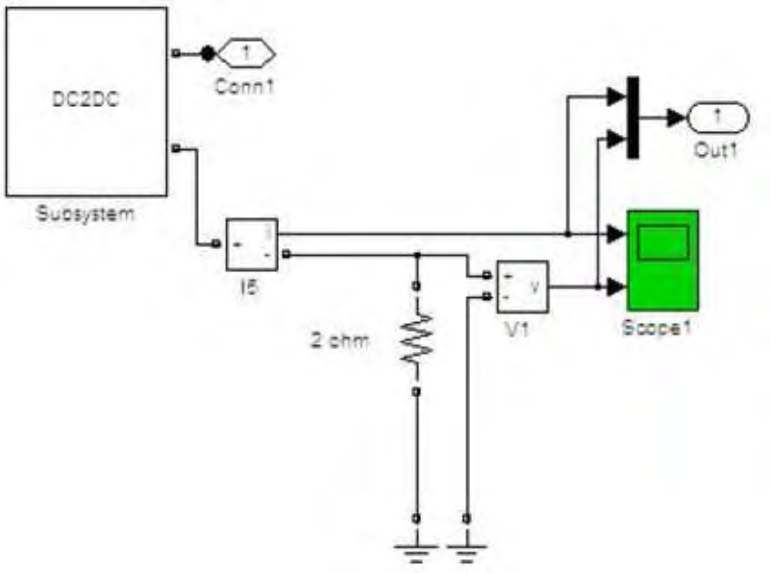


Figure A.14: Constant Power Load in the SimPowerSystems Toolbox.

APPENDIX B

MAS Negotiation algorithms

In Section 3.6.4 we presented the flowcharts of the negotiation schema. In this section we present the pseudocode of that process. The decision algorithms are used by the Initiator to evaluate the received proposals and by the Responder to generate a proposal.

B.1 Pseudocode for Contract Net Initiator

```
If SSCM has a fault and does not have other bus-connection available
then
   ZoneAgent sends a CFP message to its neighbors
   ZoneAgent's

   Wait for proposals until all neighbors answered or time is out

   If all neighbors answered or time is out then
      If proposals are different from null
         For each bus-connection requested in the CFP do
            /** All this methods only selects enough proposals to
            obtain the amount of resources the zone needs. **/

            Sort proposals by cost
            If power offered by proposals without load_shedding is enough then
               Select these proposals
               /** A proposal with load_shedding is considered
               with the power offered without load shedding **/
            Else
               Sort proposals by priority and cost
               Select the proposals needed to get the power required
            End If
```

```
            Calculate cost-path per bus of selected proposals

            /** If one or more of the selected proposals need to do
            load shedding the total cost is incremented based on
            the load-shedding-priority cost**/

            Add to the cost per bus the load-shedding-priority cost
         End For

         /** If the zone is acting as a bridge (inner Contract Net
         Initiator) the next step is skipped, because the ZoneAgent
         needs to build proposal for each bus-connection.**/
         Select best bus-connection proposals based on total cost

         For each proposal do
            If the proposal was accepted
               send ACCEPT_PROPOSAL message
            else
               send REJECT_PROPOSAL message
            End If
         End For

         Perform Tasks
         Establish the control actions based on the selected proposals
         Send control actions to CommunicationAgent
      End If
   End If
End if
```

B.2 Pseudocode for Contract Net Responder

```
If "call for proposal (cfp)" message is received then
   If Zone has enough power available then
      create a proposal
   else
      If Zone has available neighbors then
         Create an inner ContractNetInitiator and send a CFP
         for the resources the Zone can not supply
         Follow the process of ContractNetInitiator
         Create a proposal based on the selected proposals of
         subcontractors and its power available
      End If
   End If

   If proposal is null then
      Get the information of the clients
      Select the clients with less priority than the zone
      who sent the CFP to get the power required
```

```
      If power is enough then
         Save the information of clients to disconnect
         Create a proposal based on the power available and the power
         of the disconnected clients.
       End If
   End If

   If proposal is different from null then
      Send PROPOSE message to the initiator
      Wait for confirmation message

      If proposal is accepted then
         Perform tasks
         If the proposal has clients to disconnect
            For each chose client do
                Send a DISCONFIRM message to disconnect the client
                If the client-proposal had subcontractors
                   Send a PROPAGATE message to update the
                   information of client
                End If
            End For
         End If
         Establish the control actions based on the selected proposals
         Send control actions to CommunicationAgent

         If process was successful then
            send INFORM message to initiator
            send CONFIRM message to subcontractors
         Else
            send FAILURE message to initiator
         End If
      End If

   Else
      Send REFUSE message to the initiator
   End If

   End If
End If
```