Thesis

Synchronized Real-Time Simulation of Distributed Networked Controls for a Power System Case Study

Submitted by

Abhishek Jain

Department of Electrical and Computer Engineering

In partial fulfullment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2013

Abstract

Synchronized Real-Time Simulation of Distributed Networked Controls for a Power System Case Study

The purpose of this study is to develop and implement a distributed networked control framework for a power system simulation. The study addresses and improves upon speed and accuracy of simulation for computationally intensive power system dynamic simulations and distributed control utilizing Hardware-In-Loop (HIL) simulations. A dynamic four-bus test-case microgrid simulation is first constructed using SimPowerSystems^{TM} toolbox of Matlab^{TM} with renewable energy penetration. Parallel processing is achieved using a discrete real-time simulator *Opal-RT* by distributing the computation among its various processors and thus achieving real-time performance.

Maximum power point tracking (MPPT) controls for various photo-voltaic (PV) systems are distributed among external simulation platforms with the use of a client-server communication architecture and application layer messaging network protocols. The various networked platforms implementing control algorithms include general purpose and data-flow graphical programming languages. The solar irradiance profile for various PV systems is generated from an external spreadsheet data source as another networked module. Also included in the communication network is a commercial off-the-shelf (COTS) controller - a substation automation platform *OrionLX* which is used for supervisory control of the various relays in the microgrid feeder simulation.

Finally, a case study is presented which involves all of the above mentioned components - MPPT control and irradiance profile generation for PV systems as well as fault isolation in a microgrid using HIL supervisory relay control - as distributed elements of a communication network with the real-time server. Modbus TCP/IP is used as the networking protocol while the networked control platforms are developed in C# and Simulink^{TM} programming languages. Performance and bandwidth of the interdisciplinary system are analyzed. From the results of this study, it is concluded that the combination of a parallel processing and distributed control approach can be an effective strategy for improving dynamic power system simulations.

Acknowledgements

First and foremost I would like to thank my adviser Dr. Peter Young for his guidance in my thesis work and his amazing lectures in *Control Theory* which have inspired me to continue my graduate studies in that area of expertise. His humble attitude and helpful nature made my first steps in academic research a fulfilling and exciting experience.

I would like to thank Prof. Dan Zimmerle not only for his technical guidance regarding my thesis but also for the opportunities he provided me which made this multidisciplinary thesis work possible. I am grateful for the industrial research exposure I have received in the last two years as his student.

I would like to thank Mr. Jerry Duggan for guiding me in the software development process of my work. His patience and expertise have played a crucial role in my becoming a better computer programmer and in the completion of this thesis.

I would also like to thank Dr. Siddharth Suryanarayanan for his valuable comments and suggestions. Special thanks is given to Dr. Edwin Chong for his motivational (and highly enthusiastic) talks on academic research, which have inspired me to pursue a research-intensive career.

I would like to thank my friend and colleague Yi Han for his collaboration on my first paper and his invaluable counsels on the intricacies of graduate school. Finally, I thank my family for their love and support.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

1.1. Motivation of the Study

Problem Statement: Power system simulations play an important role in the study of stability and performance of electrical power systems, for instance load flow analysis and transient stability analysis. It involves modeling of power system equipment, integration of conventional and renewable sources into the power grid, and implementing improved control strategies for regulating the various dynamic and steady-state parameters in a power system. The last few decades has seen the US electricity transmission system face many challenges, including deregulation of the electricity grid, heavy integration of renewable energy sources and storage systems and utilization of fast-acting power-electronics based control equipment like FACTS compensation devices. Increasing demands on the transmission system has stretched the electrical performance to its system tolerable limits [1]. As a result, the complexity of (dynamic) power system simulations is increasing and improved novel approaches have to be implemented for analysis of power systems considering important factors like accuracy and speed of the simulation [2]. Some of the related topics discussed and implemented in this thesis with regards to a more efficient dynamic power system simulations are as follows:

Parallel and Distributed Processing: The different components in power system simulation models are usually made up of non-linear ordinary differential equations, which are solved by the modeling packages, for instance SimPowerSystems toolbox of Matlab. These equations can interact heavily with other sets of equations (for other components) in the system, resulting in a much more complicated system with a multiple of states. Once modeling is achieved, a thorough analysis of a practical system will generally require a large number of simulation runs. If these runs prove too much for one single processor (simulation time is very slow), the model is usually simplified by approximated low-order equations. This approach sacrifices accuracy of the model and thus can produce impractical and/or inefficient results [2].

Using parallel processing of these simulation models can help preserve important system dynamics like transient and small-signal stability. Parallel processing involves dividing the simulation into multiple hardware components to perform the simulation as one single simulation [3]. In our study, we have used a discrete real-time simulator ‘Opal-RT’ for dividing up our power system simulation in 12 processing cores. This enables the real-time simulation required for hardware-in-loop analysis of physical system components. As described in [2], the measure *speedup* denotes the actual advantage (gain) when moving a simulated model to a parallel computing multiprocessor. These analysis are given in Chapter 6 and was observed that *speedup* for our study is significant.

A distributed system on the other hand consists of a number of independent computers connected via a communication network. These computers (or processors) are given specific individual tasks to solve one common problem as a single unit. The communication network can be composed of anything from Ethernet to fiber optic cables. Some of the advantages of a distributed computing system as compared to running an application on a single processor are reduced cost, freedom of model expansion (under the limit of communication bandwidth) and increased configurability [4]. In our study, multiple autonomous computers are used as various ‘clients’ in a client-server communication model for distributed control over the primary simulation components. The data packets are exchanged in real-time. Since the server side of the above mentioned client-server model utilizes a parallel computing multiprocessor for accurate real-time simulations, both parallel and distributed processing advantages are implemented for system improvements.

HIL Testing of Commercial Equipment: Distribution protection and automation devices are required to undergo a thorough testing procedure before they can be implemented with the real grid [5]. For our study, a substation automation platform 'OrionLX' from NovaTech LLC is used to isolate a three-phase fault in a simulated microgrid system. It is a processor in itself and hence is able to make high-level decisions for various intelligent electronic devices (relays) in the system. This allows us to monitor the performance and limitations of the automation system. We can also analyze the efficiency of its programmed logic in special case scenarios, for instance large renewable energy penetrations to the grid and the response of the controller. Using a real-time simulator gives an added advantage of performing Hardware-In-Loop (HIL) simulations over the same communication network as the distributed system [5]. In essence, the hardware-in-loop device becomes a part of the distributed system architecture and can be regarded as just another autonomous computer system exchanging data with the server simulation.

Limitations on Computational Style: Simulation packages for dynamic power system simulation like Simulink/SimPowerSystems use a variety of solvers for ordinary differential equations. The user also defines a specific time-step approach for the whole system [6]. For a system with continuous states, a continuous solver is used and for a system with discrete states, a discrete solver is used. The user selects a fixed step discrete solver or a variable step discrete solver, depending on the model. Any one of the various integration techniques can be used for the solvers [6] and it is not unusual to find that one technique performs better over others for a specific part of the system. This problem is solved by using distributed computing for different parts of the system. For instance, in our study, the incremental conductance Maximum Power Point Tracking (MPPT) controller for a specific PV system is running in variable step while the PV system itself runs on fixed step-size, while both being components of the same system. Also, the input data for a system can be treated as yet another distributed computer (though there is no feedback loop here). Instead of porting input data from source software to a format recognized by the simulation, the data can be directly fed through the communication network directly from the source software. For instance, a wind profile data generation software can deliver data packets directly to the input of a simulated wind turbine system through the networked communication.

Model Environmental Characteristics: Since the objective of power system simulations is to match the response of the real system as closely as possible, all aspects of the system dynamics have to be considered for an accurate simulation. In most physical systems, inter-device communication takes place through small communication networks themselves [7]. These networks are not usually modeled when creating a simulation of the system. Distributed processing facilitates modeling those networks in order to analyze scenarios like node failure or network failure between two points. For slow-speed networks in real systems, we can intentionally introduce latency to correctly simulate the behavior of the system. In our study, an intentional latency of 1 sec is introduced between the Photo-Voltaic (PV) system and the MPPT controller to simulate the real system.

1.2. Proposed Interdisciplinary System

The proposed interdisciplinary system, addressing all of the topics discussed above, has a distributed networked control architecture (Figure 1.1) which serves as a framework for Cyber-Physical Systems (CPS) and Networked Control Systems (NCS). As shown in the fig below, the client-server communication network is implemented with Modbus TCP/IP (Transmission Control Protocol/Internet Protocol) standard protocols. The discrete real-time simulator *Opal-RT* serves as a central node (server) for the communication network while the other nodes (clients) are networked distributed controllers. Every client or the single server has a networking wrapper/layer on top of it to facilitate communication. Each of the nodes in the system has a different IP address as indicated in the figure.

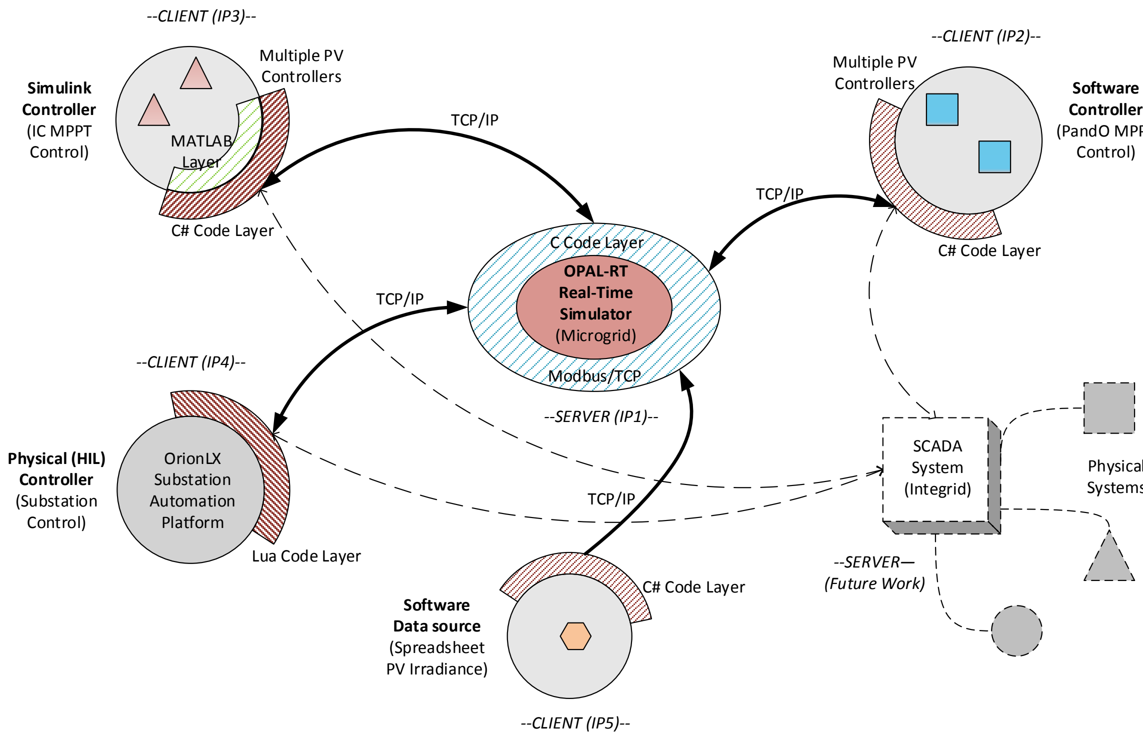


Figure 1.1. Distributed Networked Control Framework for a Microgrid

The Server: The Opal-RT server has a layer of C code which contains socket programming as well as Modbus/TCP implementation. Socket programming is implemented such that the server can communicate with multiple clients simultaneously. A full dynamic microgrid power simulation is running on the Opal-RT system in real-time with its multiple devices communicating with the distributed clients for control data. For instance, a photovoltaic system running on the server requests control data from the MPPT controller on a distributed server. The PV system and the MPPT control are in a feedback loop through the network. Other devices like overcurrent (OC) relays, wind turbine generation systems are also being controlled. While not in a feedback loop, the photovoltaic system is receiving its input solar irradiation data from one of the clients as well. The data is time-stamped for analysis and proper execution. The server is given IP1.

Clients: The various distributed clients are networked control devices for the primary simulation running on the server (Opal-RT). The control algorithms are all executing on their native platforms on their host computers. Each of these nodes also has a software layer for implementing network communication. The *software controller* node IP2 implements algebraic algorithms in C# language. The *simulink controller* node IP3 implements simulink & stateflow dynamic controllers in Matlab programming language. Matlab-.NET interfacing is done to load the relevant .NET libraries in Matlab. The *physical controller* node IP4 is the substation automation platform implementing relay control algorithms in *Lua* programming language. The nodes IP2 and IP3 each have a software layer implemented with the help of .NET libraries, written in C#, while the commercial control device OrionLX (IP4) has its own in-built software layer for client-server communication.

1.3. Organization of Thesis

The thesis has been organized as per the following chapters:

- **Chapter 2:** This chapter presents the literature review for the proposed system. The study presented in this thesis is largely experimental with theoretical concepts and referred analytical results taken from the noted references.

- **Chapter 3:** This chapter examines the power system modeling done in SimPowerSystems/Simulink of Matlab for the case study presented in Chapter 6. Modeling of various components in the microgrid power system is explained in somewhat detail and analysis is done in Matlab to verify their correct behavior. These components are then integrated to form a single four-bus feeder serving as a microgrid to the utility grid. Performance of two of the most widely used MPPT controllers is also shown.
- **Chapter 4:** This chapter describes the system architecture of the proposed system. An overview of commercial devices like *Opal-RT* and *OrionLX* is given with explanation on their working and their role in the project. Also explained is the role of distributed computers used for networked control of renewable energy sources in the microgrid.
- **Chapter 5:** This chapter explains the software architecture for the distributed system. With the help of figures, the client-server networking methodology and its implementation in the current context is explained. The architecture of the server programming layer is also given. This chapter studies the implementation of various distributed control platforms for the system. First the structure of the .NET programming layer, for client-server communication, is explained. Then the implementation of control algorithms by interfacing software class libraries and Matlab is explained.
- **Chapter 6:** This chapter gives a case study for visualizing all the interdisciplinary aspects of this thesis in one practical application. Fault isolation, in case of a three-phase overcurrent fault, is done in the microgrid supported by distributed controller

supervisory logic. Analysis is done on the performance of the built system and limitations are monitored and noted. Robustness of the system is analyzed by increasing the number of networked clients and monitoring response times. Latency issues and bandwidth limitations on the network are noted.

- **Chapter 7:** The final chapter highlights the important features of the distributed control system framework and the scope of future research. Along with developing a groundwork for CPS, this thesis addresses the performance aspects of the developed system. However, improvements to the system in future work are required as detailed in this chapter.

CHAPTER 2

Literature Review

This chapter reviews the current trends of the various components, methods and ideas implemented and discussed in this thesis. The first six sections of this chapter cover a brief literature review of all the interdisciplinary aspects of this work. The final section correlates the study in this thesis with the general literature as described in previous sections of this chapter.

2.1. Distributed and Parallel Processing

For dealing with large-scale and computationally intensive power system problems, distributed and parallel processing is proving to be one of the most effective new developments [3, 4, 8]. The two forms in which parallel processing has been widely implemented are: multiprocessors and multicomputers [9]. A multiprocessor consists of various processors sharing a common memory while multicomputers use message passing to communicate among its various processor-memory pairs [2]. The various advantages of distributed computing over conventional approaches are given in [8]. In power system applications, parallel and distributed processing have impacted several areas such as real time control for voltage stability assessment and optimal power flow [10, 11] and developing real-time simulators for accurate dynamic power simulations and design/testing of new equipment [12, 13, 14]. Several parallel and distributed techniques/algorithms as applied to power systems have been reviewed in [2]. One major advantage of parallel processing has been in the development of real-time simulators. An overview of parallel discrete event simulations and timing synchronization is given in[15]. Commercial real-time simulators like RTDS (Real-Time Digital Simulator) and Opal-RT system have been successful for industrial and academic research purposes [16].

2.2. Hardware-In-Loop

Hardware-In-loop simulation connects a physical controller is connected to a simulated plant, which is executing on a real-time platform [16]. This technology is thus a direct consequence of both parallel and distributing processing. Some of the case studies involving real-time HIL implementations include:

- Digital controller testing using Real-Time Virtual Testbed (RTVTB) [17].
- Parallel simulation of power drives and electric circuits using Opal-RT discrete real-time simulator [18].
- Simulation of distributed intelligent energy management systems for microgrids, using intelligent agents and zigbee wireless communication protocol.
- Power system modeling for active compensator and realy HIL tests, using National Instruments PXI controller 8196 as a digital simulator [19].

Another technique for testing of physical devices with simulations, derived from HIL, is called the Power Hardware-In-Loop (PHIL). While HIL can only be used for small-signal control applications while PHIL involves actual power transfer from to/from the hardware being tested [20]. PHIL is accomplished by the means of a logical partitioning of the full system into Hardware Under Test (HUT) and Rest of the System (ROS). This methodology decouples the physical device from the base simulation [20]. Several PHIL applications - for instance high-speed generator testing, fault current limiter testing and ship propulsion drive testing - have been discussed in [21, 22]. Differentiation and comparison between Controller Hardware-In-Loop (CHIL) and PHIL experimental studies have been given in [23]. Protection systems equipment simulation and fault isolation studies have been studied extensively in the context of HIL tesing [24, 25, 26].

2.3. NCS and Cyber-Physical Systems

A Networked Control System (NCS) is a feedback control system wherein the interface between the plant and the controller is built via a network communication system. With the increase in complexity due to the network, several NCS issues have been addressed in the literature [27]

- Network-induced delay while exchanging information can degrade the performance of overall system [28]. Delay compensation techniques like gain-schedular middleware [29], predictive modeling of NCS [30] and Linear Matrix Inequalities (LMI) minimization [31] have been studied.
- Finite amount of available bandwidth can limit the number and size of input/output control signals. Bandwidth allocation and scheduling strategies for NCS have been studied in [32, 33, 34].
- Network security in wireless NCS is an issue as these network mediums are highly susceptible to easy interceptions. While taking care of network security, the trade-off between security and performance of NCS should also be considered. Security protocols such as wired equivalent privacy (WEP) and extensible authentication protocol (EAP) have been implemented in [35, 36] to address these issues.

A few of the various case studies involving NCS are:

- A multi-sensor network-controlled navigation system for multi-robots, using a novel concept known as iSpace has been developed at North Carolina Sate University [37]. Sensor data is fed through cameras to the networked controller, which then gives out control signals to the robots for obstacle avoidance and other navigation objectives.

- A supervisory load-frequency control (LFC) strategy for multi-area power systems using a Distributed Reference Offset Governor (DROG) device [38]. The latency due to the network is modeled by a time-delay in the simulation.
- A water process remote monitoring network system in a firepower plant using Modbus TCP protocol [39].

The above described networked control systems, coupled with real-time (power) hardware-in-loop simulations, can potentially be seen as a basis for an interdisciplinary CPS where the physical system (plant) is combined with multiple networked software modules (controllers) over a communication network. A model-integrated development approach to cyberphysical systems (CPS) is explained in [7], where the authors recommend an early simulated-to-physical world integration approach for model-based design. A review of developments in the Cyber-Physical Energy Systems (CPES) [40] emphasizes power systems/applications (especially smart grids), keeping in mind optimal power flow and energy control. Control of the microgrid is proposed to be achieved via a Microgrid Central Controller (MGCC) which is interfaced with real-world physical components of the grid - for instance PV systems and conventional generators - via a communication network [40].

2.4. Multi-Agent Systems for Microgrid Control

The next logical progression in distributed controls, as related to power systems, has been the development and implementation of Multi-Agent Systems (MAS), especially for microgrid control [41, 42]. An overall review of MAS utilized in power system applications is provided in [43] . According to [44, 45], MAS technology can be the basis of an organized control strategy where the full microgrid control is split among various intelligent distributed local controllers. The distributed technology has been evolved with new capabilities for more complex microgrid controls [46]. MAS for wide-area control of power systems and single-machine systems has been studied in [47, 48, 49, 50]. A MAS can be either a physical hardware system or a simulated virtual one.

The control of distributed energy sources in a microgrid by using control, management and ancillary agents, has been proposed in [51]. The authors also use a Microgrid Central Controller (MGCC) for managing various agents in the system. Similar to the study in this thesis, the HIL technology with the Opal-RT simulator has been used for the implementation of physical agents. In [52], authors present another variation to the MAS technology, i.e., Intelligent Distributed Autonomous Power Systems (IDAPS). The focus is on customer-owned DER units which may or may not belong to different utilities. Implementation of the IDAPS concept is realized in [53] where SimPowerSystems toolbox of Matlab is used to simulate microgrid hardware with integrated agent control. TCP/IP protocols are used for software-simulation-hardware communication.

2.5. Synchronized Control and Network Limitations

For networked control of simulated and/or physical devices, synchronization is required between the plant and the controller over the transmission network. This introduces two new factors: time delays and response times, which have to be considered for optimal performance and stability of the networked distributed system [54, 55]. Synchronized control for distributed generators over an IP network from synchrophasor measurements is shown in [54]. Response-time analysis fora private network connected IEDs in a system is done in [55]. Both these works show independent studies on private networks and hence a case study for response time analysis on a LAN network in needed.

2.6. Modbus/TCP in Networked Control

Modbus is a request/response distribution automation protocol widely used to control industrial devices over a network [17]. Hence industrial devices for power distribution already have Modbus interface built-in and thus the wide use of Modbus as a communication protocol for HIL simulations seems a logical choice. The basic structure, with framing and error checking techniques, is given in [56]. For Ethernet compatible equipment, TCP/IP protocol can be integrated into Ethernet to form the physical and data link layers of the OSI model [39]. In this study, Modbus TCP protocol has been used for networking distributed clients with the real-time server. Modbus/TCP has been used a master-slave protocol in various industrial and academic applications:

- In [57], Modbus/TCP was used for the client-server communication and interfacing between different controls equipment of the SCADA control system for analyzing its security.
- The protocol was implemented for the HIL testing of power electronics equipment using a multi-threaded server model for Modbus [17].
- In [58, 59], it is shown that Modbus/TCP can be used effectively for the IED architecture. IED's provide various functions in power system industry like protection, monitoring, control and metering.
- The protocol can also be used in a NCS due to its simple and wide usage and low latency [39].
- The protocol was used for communication between two separate real-time simulators, the RTDS system and the Opal-RT system, for a combined electro-thermal simulation framework as shown in [60].

2.7. Summary

The literature review describes the current trends in the interdisciplinary field of distributed controls, as applied to power system distribution networks (including microgrids) and its simulations. This thesis presents a case study of the proposed hardware-in-loop distributed controls set-up for a microgrid and takes various theoretical concepts from the noted references in this chapter. Besides providing a novel experimental framework for CPSs, this study also indicates which type of power devices can be a part of this framework as per the response time analysis done in this study and also the stability of the system due to delays induced because of the network latency.

CHAPTER 3

Modeling of Microgrid System

A *Microgrid* is a localized power system, which views generation and associated loads as a subsystem [61]. Microgrids generally contain multiple distributed generation sources and can operate in both islanded and grid-connected modes. The distributed generation in a microgrid can include renewable sources of energy like photovoltaic systems, wind turbines, fuel cells etc. Microgrids can thus be a highly reliable and green electric power source.

In this chapter, a four-bus, 25 kV transmission-level microgrid [1] system is constructed with high renewable energy penetration in the form of multiple PV systems. Protection equipment is simulated in the form of overcurrent relays and autorecloser circuit breakers. The simulation is carried out with the help of SimPowerSystems^{TM} toolbox of Matlab^{TM}. In subsequent chapters, this constructed system is then used as the 'plant' while many of the control elements are implemented as 'distributed controls' for a case-study analysis of networked distributed control mechanism in a simulated microgrid.

A one-line diagram of the constructed system is shown in Figure 3.1. The voltage source *S1* provides a three-phase AC constant voltage of 25 kV to the microgrid. The PV systems are connected to the AC microgid transmission system through inverters *Inv(1-4)* and step-up transformers *T1* and *T2*. A three-phase overcurrent fault is placed between Bus 2 and 3 for the four-bus feeder system. Control of the peak power of the PV systems is achieved via two of the most widely used MPPT control algorithms:

[1]It is to be noted that the 'Microgrid' system constructed in this study is based on assumptions and parameters so as to simplify the power system operation (but not its dynamics) and hence does not represent real-world scenarios or current industry standards. This is done so as to keep the focus of the study on developing the communication framework (explained in later chapters), rather than the power system operation accuracy.

- Perturb and Observe Method (*P&O*)
- Incremental Conductance Method (*IC*)

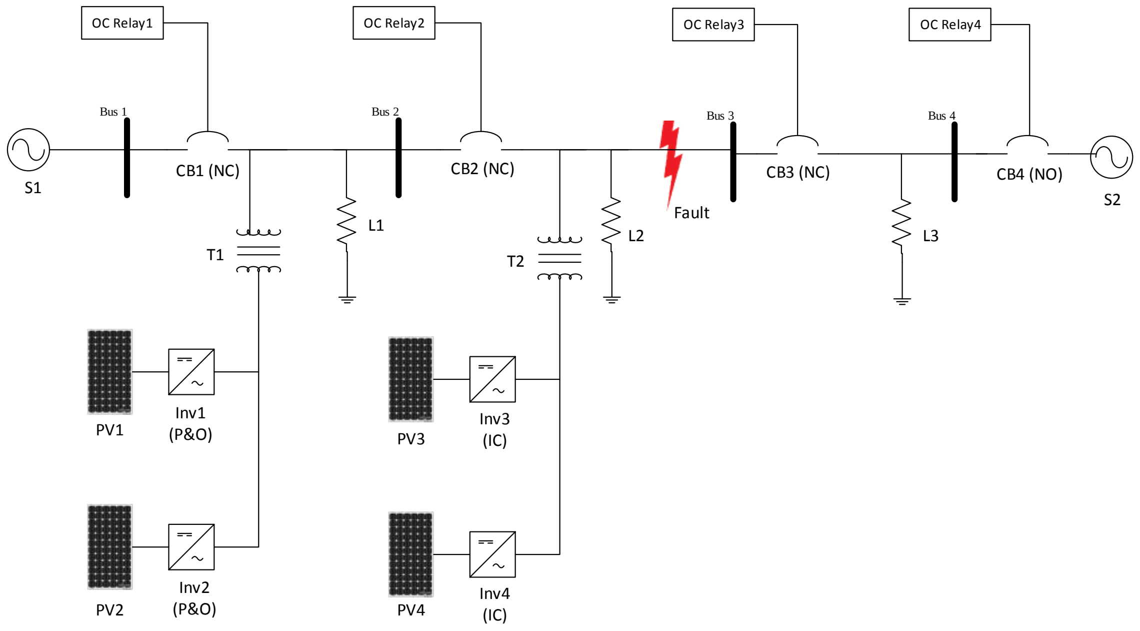


Figure 3.1. One-line diagram of a 25 kV Microgrid System

The simulated protection equipment involves autorecloser circuit breakers *CB(1-4)* and overcurrent relays *OC Relay(1-4)*. Both balanced and unbalanced resistive loads of various power ratings are placed within the microgrid feeder line. A three-phase balanced fault, between buses 2 and 3, is simulated for analysis of the protection system. A back-up generation source *S2* is also provided for fault isolation study in Chapter 7.

Table 3.1. Simulation Specifications

Parameter	*Value*
Nominal Voltage	25 kV
Nominal Grid Frequency	60 Hz
Sample Time	50 μsec
Simulation Type	Discrete (Tustin)
Solver Type	Fixed Step ode3 (Bogacki-Shampine Formula)

The simulation parameters are as given in Table 3.1. Fixed step-size is used so as to make the simulation compatible with the real-time simulation environment (*Opal-RT*) [16] used in the subsequent chapters. A detailed description of the modeling of various individual components in the microgrid is as follows:

3.1. Photovoltaic Systems

Figure 3.2 represents a typical block-diagram configuration of a three-phase grid-connected PV system [62].

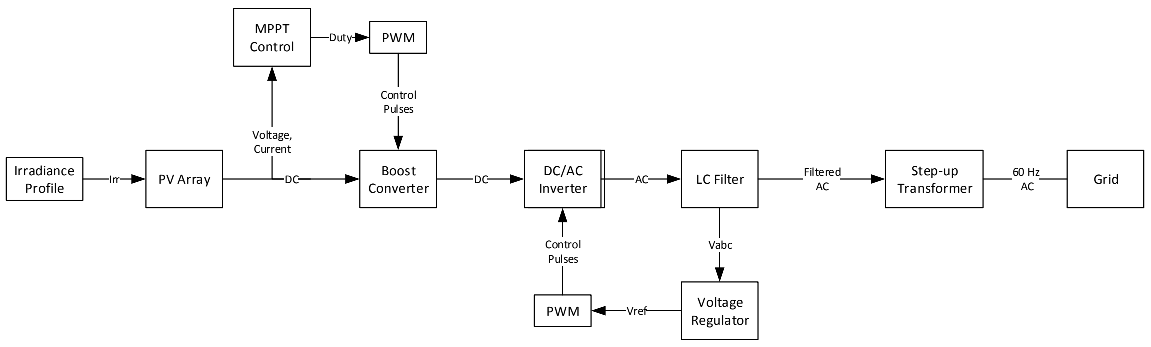


Figure 3.2. Block diagram of a Grid-connected PV System

The PV system shown consists of the following main components:

- A PV array with solar irradiance profile as an input and DC supply as output.

- A DC-DC converter which delivers the maximum power DC supply with MPPT control signal and PV DC supply as an input.
- A DC-AC voltage source inverter which converts the DC supply to three-phase AC with the help of a voltage regulator.
- An LC filter to remove undesired harmonics from the fundamental frequency.
- A step-up transformer with a breaker so the PV system can be connected to the main grid.

Each of the subsystems is modeled and analyzed in SimPowerSystems and explained in some detail in the following sections.

Note: The only difference between the four PV arrays shown in Figure 3.1 is the use of different MPPT algorithms (and/or its parameters) for individual arrays.

3.1.1. PV Array. A PV array is composed of many series and parallel connected PV modules, whereas a single module consists of a number of series connected solar cells. The basic equation from the theory of semiconductors [63] that mathematically describes the I-V characteristics of an ideal cell is:

$$I_d = I_{sat}[e^{V_d/V_T} - 1] \tag{1}$$

where:

$$V_T = k\frac{T}{qQ_dN_{cells}} \tag{2}$$

where I_d is the diode current, I_{sat} is the diode saturation current, V_d is the diode voltage, V_T is the temperature voltage, k is the Boltzmann's constant $(1.3806503 \times 10^{-23} J/K)$, q is the electron charge $(1.6022e^{-19}C)$, Q_d is the diode quality factor, N_{cell} is the number of series connected cells per module. The light-generated photo-current of the a single module is represented by I_{ph}. This value increases as the number of parallel module strings increase and is directly proportional to the solar irradiance on the module's surface.

For a PV array with 66 parallel strings (N_{par}) consisting of 5 modules (N_{ser}) each, the Equation (1) is altered to:

$$I_{d_{array}} = I_{sat_{array}}[e^{V_d/V_{T_{array}}} - 1] \tag{3}$$

where $I_{d_{array}}$ is the aggregated diode current for the PV array, $I_{sat_{array}}$ is the aggregated diode saturation current and $V_{T_{array}}$ is the aggregated temperature voltage. These values, along with the increased photo-current value $I_{ph_{array}}$ are calculated as:

$$I_{ph_{array}} = I_{ph} \times N_{par} \tag{4}$$

$$I_{sat_{array}} = I_{sat} \times N_{par} \tag{5}$$

$$V_{T_{array}} = V_T \times N_{ser} \tag{6}$$

The specific PV panel used in this study is *SunPower SPR-305-WHT PV Panel* [64]. The model parameters used for calculations in Equations (1) - (6) are given in Table A.1 in Appendix A. Figure 3.3 shows the I-V and P-V curves of the panel, derived and simulated from equations (1) and (2). The circle marker on the plots indicate the maximum power point and the dotted plots are shown for different values of input irradiances. The irradiance value used for the analysis done in this chapter is 1 kW/m^2.

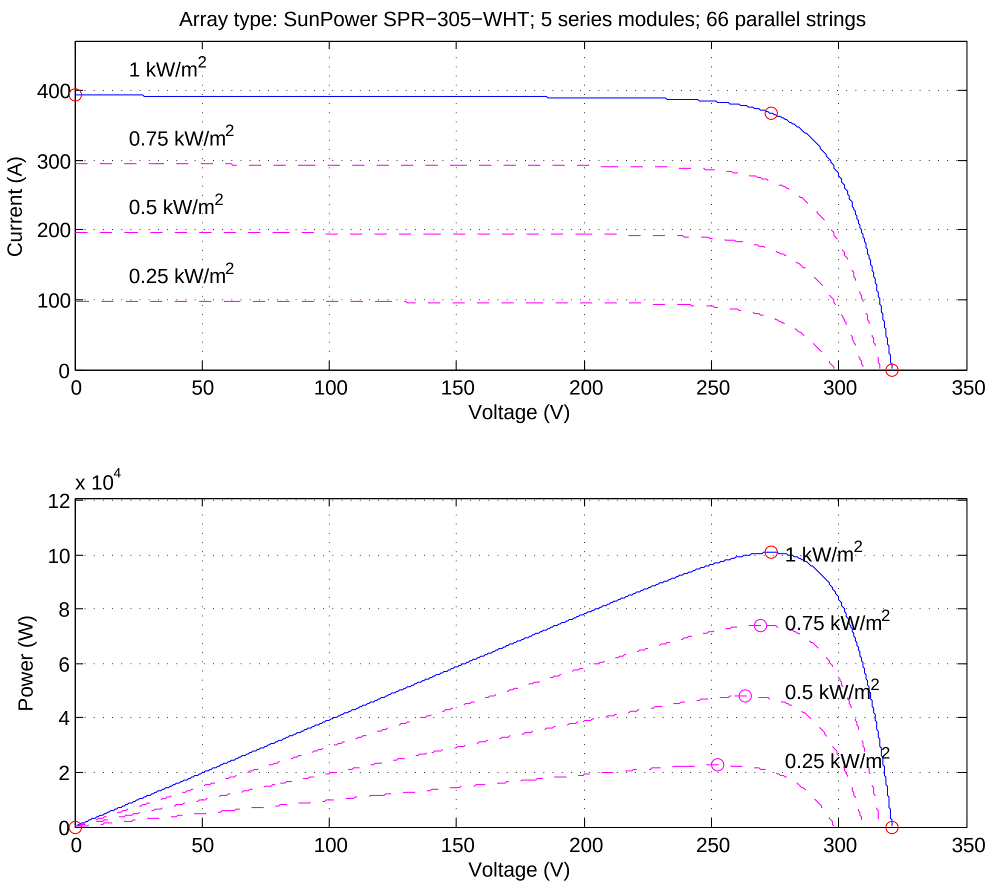


Figure 3.3. I-V and P-V curves for SunPower SPR-305-WHT PV Array

3.1.2. DC Link. The DC link for the modeled PV system consists of an initial low-pass filter, a DC-DC boost converter with MPPT control, and another capacitive low-pass filter. Referring to Figure 3.2, the primary function of the IGBT-based boost converter is to raise the output DC voltage as corresponding to the maximum power delivered by the PV array (knee of the I-V curve in Figure 3.3). It achieves this with the help of pulse-width modulated control signals (PWM switching frequency = 2 kHz) from the MPPT controller. The unmodulated control signal from MPPT is called *duty cycle* of the PV system [65]. The two "hill-climbing" MPPT algorithms used in this study are explained briefly [66] and their flowcharts/implementation diagrams are given in Appendix A in Figures A.1 and A.2.

- *Perturb and Observe Method (P&O)* [67, 68] The P&O method ensures maximum power output from the photovoltaic panel by modifying its output operating voltage or current. For instance, if increasing the voltage to a cell increases the power output of a cell, the control signal (duty cycle) increases the operating voltage until the power output decreases. Then, the voltage is decreased in small decremental steps to return to the maximum power output value. This process continues until the maximum power point is reached. Thus, the power output value of the solar panel oscillates around a specific maximum power value until it stabilizes. P&O is the most commonly used MPPT method due to its ease of implementation. One of the major drawbacks of the P&O method is that the power obtained oscillates around the maximum power point in steady state operation (as seen in Figure 3.8). Also, this algorithm has the tendency to track in the wrong direction under rapidly varying irradiance levels or temperature variations. A Matlab function is written to implement this algorithm, the flowchart for which is shown in Figure A.1 in Appendix A.
- *Incremental Conductance Method (IC)* [69, 68] The concept behind the Incremental Conductance MPPT algorithm is that the slope of the power-volatge (P-V) curve

is zero at the maximum power point, is positive at the left of the MPP and negative at the right of the MPP (Figure 3.3). The MPP for this method is calculated by comparing the instantaneous conductance (I/V) to the incremental conductance (ΔI/ΔV). Once MPP is obtained, the solar module maintains this power point unless a change in the current value occurs. This happens when there is a change in the MPP due varying ambient conditions. The algorithm then modifies the operating voltage until the new MPP is reached. This technique has an advantage over the P&O method that it can determine when MPP is reached without oscillating around this value. It can also perform MPPT under rapidly increasing and decreasing irradiance conditions with higher accuracy than the P&O method. The disadvantage of this method is that because of increased computational complexity, the MPP computation slows down the sampling frequency of the operating voltage and current. A Simulink Stateflow chart is constructed to implement this algorithm, the flowchart for which is shown in Figure A.2 in Appendix A.

3.1.3. PV Inverter.

A 3-arm bridge, with IGBT/Diodes as the power electronic devices, is used to convert the DC output of the Boost to three-phase AC. The Phase Locked Loop (PLL) within the voltage regulator (see Figure 3.2) tracks the grid voltage (1 pu) to ensure synchronization. A discrete PWM generator ($2kHz$) provides control pulses to the inverter IGBTs.

There are two sources of high-frequency noise on the inverter output voltages and currents [70] which have to be attenuated before connection to the grid (See Figure 3.4).

(1) The first one is the inverter PWM modulation frequency which is primarily attenuated by the LC filter and the transformer.

(2) The second source originates from the switching transients of the power electronic devices (IGBTs). Shunt capacitors are added to attenuate this high frequency noise component.

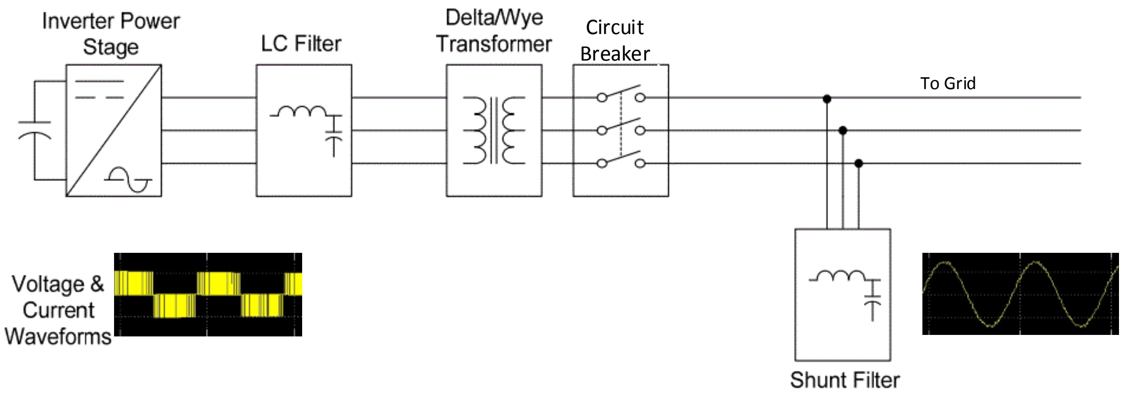


Figure 3.4. PV Inverter Filtering Configuration

A two-winding 3-phase Δ/Y step-up transformer is used with an external control circuit breaker before connection to the grid. The breaker is closed after some amount of delay (1.75 secs for PV1 and PV2, 2 secs for PV3 and PV4) so as to analyze different penetration start-times for the PV systems.

3.2. Protection System

The protection system constructed in this study consists of four overcurrent relays, with autorecloser logic, for each of the four buses as shown in Figure 3.1. Since the power system constructed is radial in nature, definite-time overcurrent relays can be used to protect the network [71]. Autorecloser logic is also built into the relays for automatic reclose mechanism of the breakers. The operation and application (modeling) of the two mechanisms (overcurrent and autorecloser) is explained briefly as follows:

- *Overcurrent Relaying*: An overcurrent relay operates or picks up when its current exceeds a predetermined value. These types of relays protects the electrical power system against excessive currents (in any of the three phases) which may arise due to any of the various types of faults or short-circuits [72]. The types of relays used in this study are *definite time overcurrent relays* which operate only when the fault exceeds a predetermined value as well as when it is persistent for a predetermined amount of time (delay). In this study, four definite-time overcurrent relays are used. The downstream coordination of the relays according to the time-delay settings is shown in Figure 3.5.

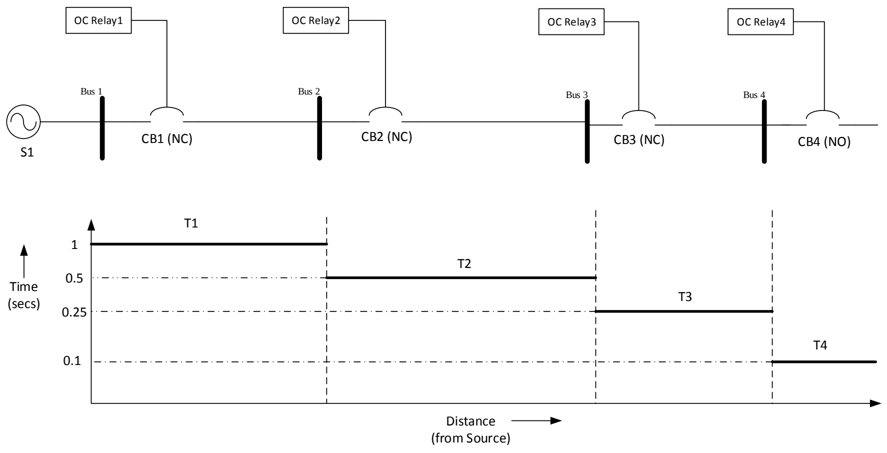


Figure 3.5. Time Co-ordination for Feeder Relays

Figure 3.5 shows that the delay time for operating each relay (T_{1-4}) has been decreased as we move away from the primary source. This is a required characteristics of relay coordination so that the relay farthest from the fault can act as back-up protection and will only operate if the primary relay fails [72, 73].

- *Autorecloser*: In physical systems, the autoreclose circuit breaker is equipped with the mechanism to automatically close the breaker after it has been opened due to a momentary fault. The control system for a recloser will allow a selected number of attempts to restore service before locking out (opening the breakers), requiring a manual reset [72]. In our study, this control logic is built into the relays itself. Figure 3.6 shows the configuration of the built relay in SimPowerSystems.

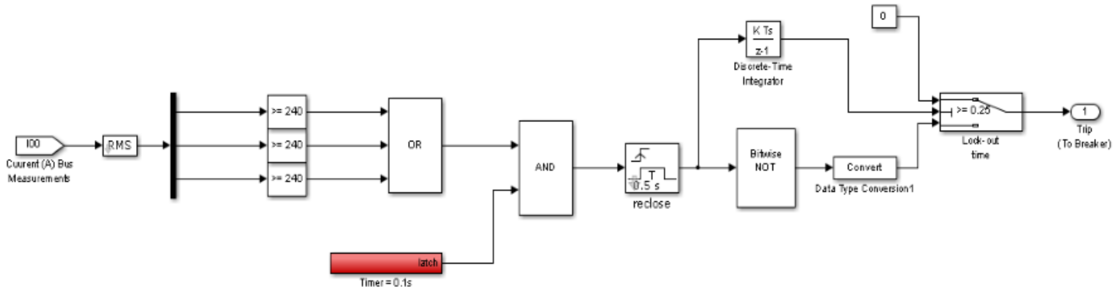


Figure 3.6. Relay Configuration

3.3. Simulation and Fault Analysis

The microgrid system is simulated in Simulink environment and relay fault protection is analyzed. As shown in Figure 3.1, circuit breakers for Buses(1-3) are normally closed while that of Bus 4 is normally open. Simulation is carried out for 5 secs and the following results are analyzed:

(1) *Bus Currents*: RMS current values are plotted in Figure 3.7 and it is seen that for a fault duration of $2 - 4secs$, relays R1 and R2 start their reclose mechanism after detecting the overcurrent fault. Since the lockout time for R2 is $0.5secs$ (See Figure 3.5) and it is the closest downstream bus to the fault, it locks out after

2.5*secs* and hence current in Bus 2 goes to zero permanently. R1 also sees the fault at 2*secs* but since it is acting as a back-up protection to R2, it closes its breaker (resumes normal operation) after R2 locks out.

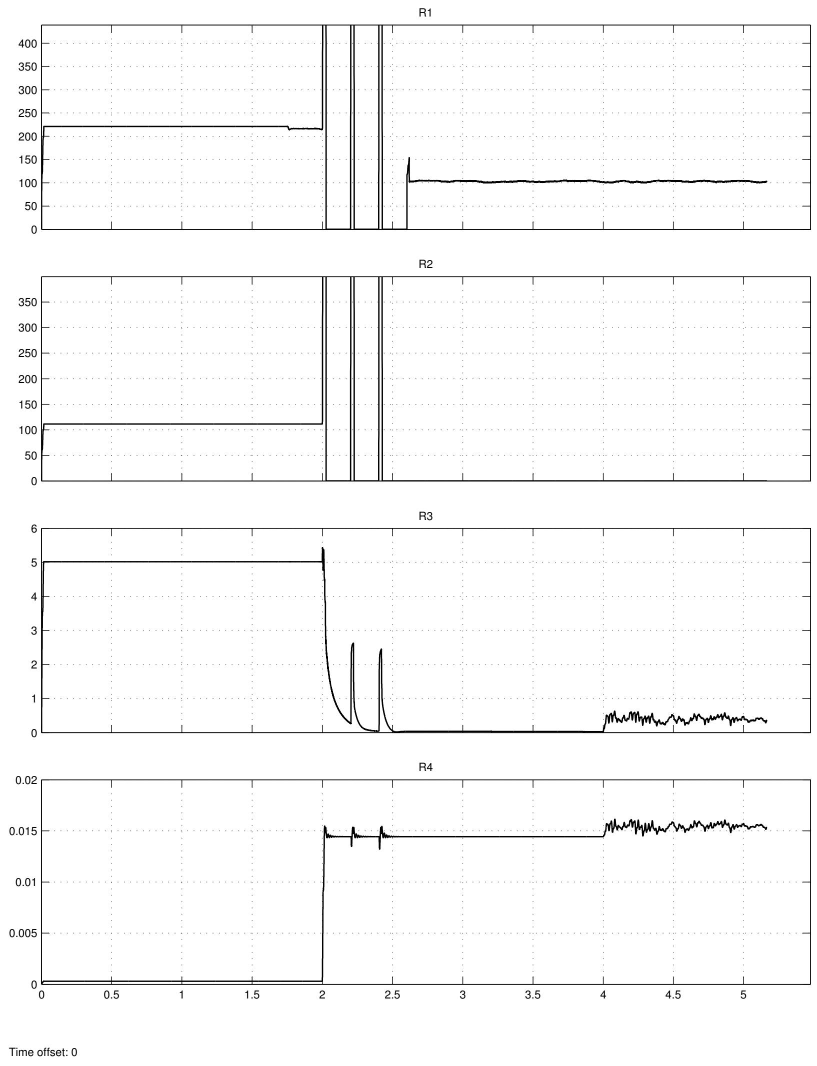


Figure 3.7. RMS Bus Currents for Relay Operation, showing time incerment (secs) on the x-axis and RMS Current values (Amperes) on y-axis

It is also seen that the current goes to zero permanently for Bus 3 after R3 locks out, even though the fault is not present in the region between Buses 2 and 3. This means that the loads $L3$ (Figure 3.1) become non-serviceable even though it is a non-fault area. Hence there is a need for a supervisory control which is addressed in Chapter 7 of this thesis. The currents in Bus 4 remain zero throughout as CB4 is normally open and no control command is given for its operation.

(2) *PV Duty Cycles*: Figure 3.8 shows the duty cycle control signals for the four PV systems.

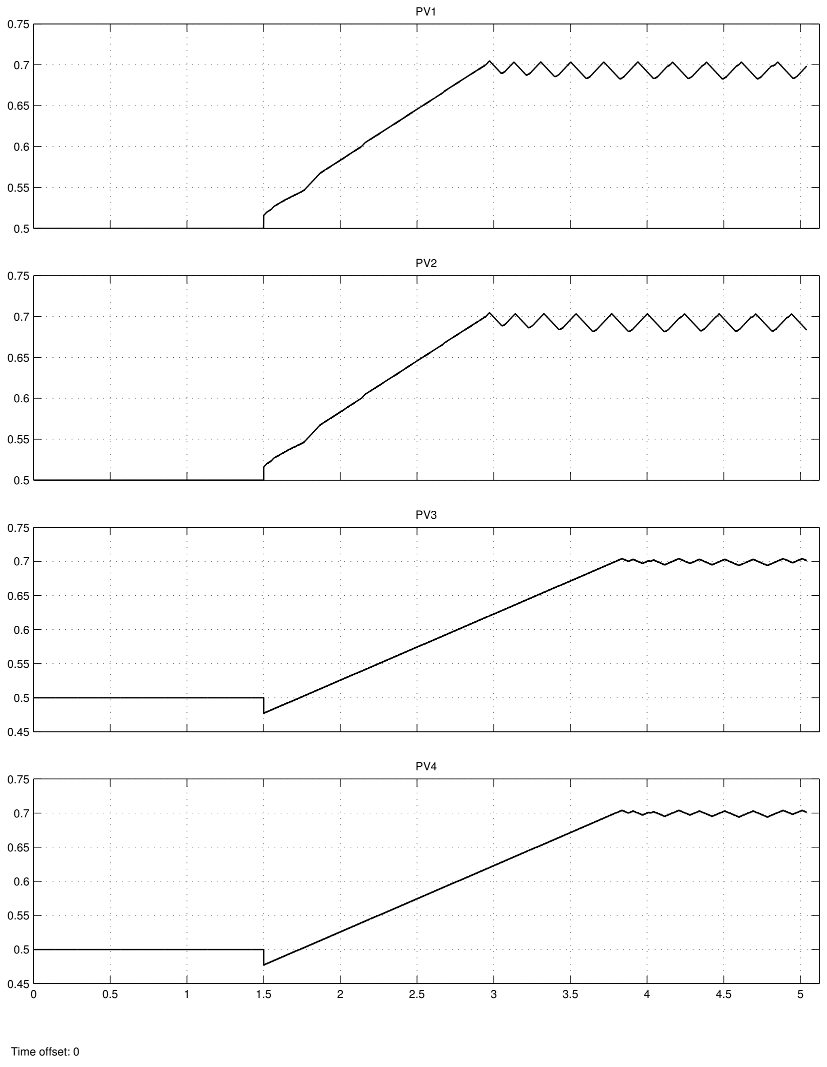


Figure 3.8. Duty Cycles for Microgrid PV Systems, showing time increment (secs) on the x-axis and the control signal 'duty cycle' value on the y-axis

PV1 and PV2 use the *Perturb* & *Observe* algorithm for duty cycle control while PV3 and PV4 use the *Incremental Conductance* algorithm. This is seen in Figure 3.8 where duty cycle plots for PV1 and PV2 are similar, same is the case for PV3 and PV4. As seen, duty cycles for both the algorithms oscillate at around 0.7 i.e. the maximum power point. A $1.5secs$ delay is introduced so that the grid is stabilized before introducing renewable integration.

CHAPTER 4

System Architecture

The interconnection of different subsystems via a communication network for the proposed system is as shown in Figure 4.1. It consists of various distributed nodes with intercommunication via Ethernet-LAN as shown. The real-time simulator *Opal-RT* acts as the server node and the rest are client nodes with either control, monitoring or data sourcing functionality. Programming and data source platforms (i.e. Simulink, C# and Excel spreadsheets), along with physical hardware equipment, have been networked as client nodes.

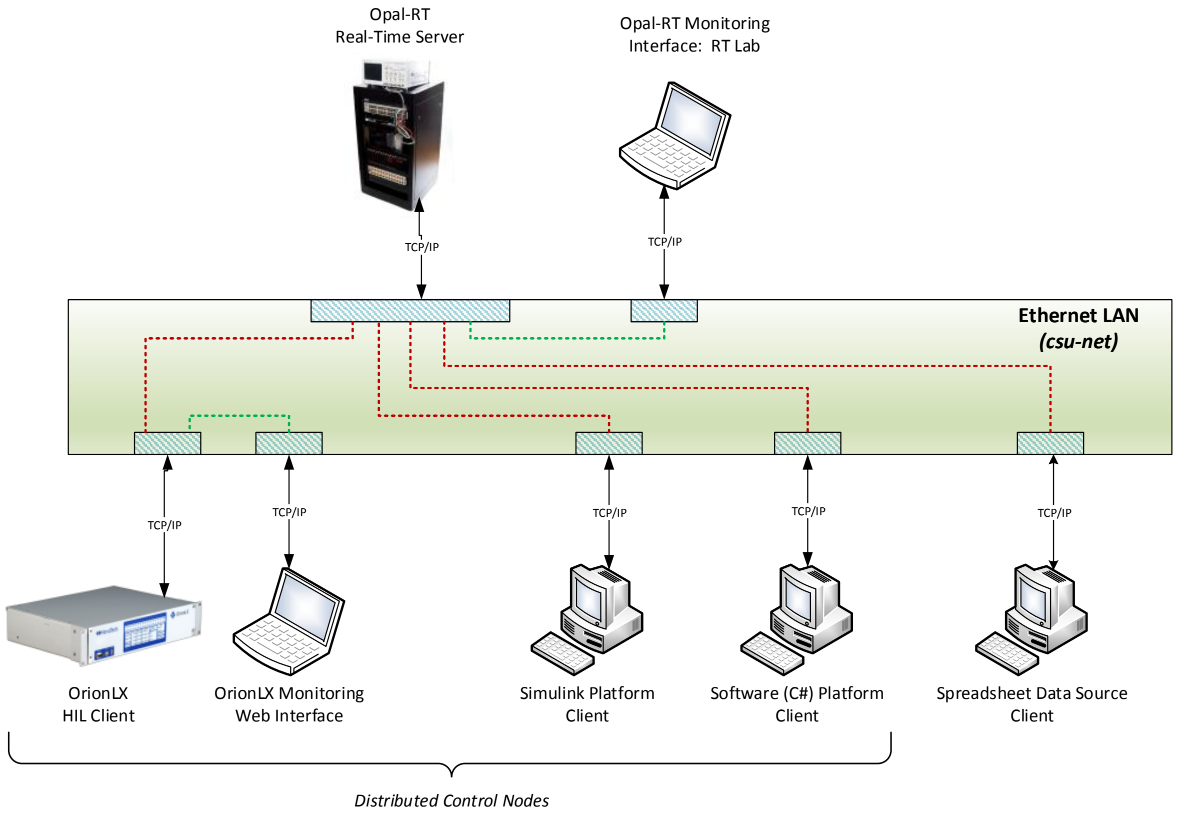


Figure 4.1. Interconnected System Architecture

Hardware-In-Loop (HIL) simulation is a technique where parts of the simulation are replaced by their corresponding physical components. Generally, a physical controller is placed in a feedback loop with a simulated plant so as to test the hardware under various simulated conditions without dealing directly with various issues, for instance time consumption, cost and safety, that would result if the controller was to be tested in the field directly. The plant is generally represented by a mathematical model of its dynamic systems [74].

With reference to Figure 4.1, in this study, the server runs the microgrid simulation in real-time, synchronized with the external nodes, even though the system on the whole has an asynchronous communication architecture. HIL simulation is achieved with the integration of commercial power equipment *OrionLX* into the system. The two COTS devices used in this architecture, *Opal-RT and OrionLX*, are accompanied by their respective Human-Machine Interface (HMI). A client-server communication model is used for the control devices polling simulated power devices on the real-time server. A multi-level software and network architecture model for the system is explained in detail in the subsequent chapters. The purpose and functionality of individual nodes is explained in the following sections:

4.1. Real-Time Server: Opal-RT

Opal-RT^{TM} (model OP5600) is a discrete real-time simulator with 12, 3.3-GHz processor cores. The single target system uses the Red Hat Linux real-time operating system. The Opal-RT system is capable of simulating slow and fast transients of a large power system effectively. The real-time capability of the simulator makes it ideal for HIL prototype testing [75]. It separates the original Simulink model into subsystems to deploy on its various processors. Each portion of the model is coded in C language and 'built' for execution among its processors. After compilation, the code is loaded into the target hardware and executed via parallel processing (the full compilation-loading-execution process is described in detail in [76]). Opal-RT has an HMI called *RT-Lab*, a real-time simulation software environment which provides the user tools to develop and execute models written in the Matlab/Simulink environment on the Opal-RT hardware. Communication between the target hardware and the HMI console is achieved through a TCP/IP connection.

In our study, the Simulink microgrid model (constructed and analyzed in Chapter 3) is first configured into subsystems for compilation by RT-Lab. It is then converted to C code by *Simulink Coder* and loaded into the Opal-RT hardware. The display blocks can be viewed in a console window when the model is executed. The relevant software (written in C) is also loaded onto the target before compilation. These codes are used to send/receive data packets to/from the external distributed controllers with the help of TCP/IP protocols in our application (discussed in subsequent chapters). The internal communication of the simulated model and software is done via shared memory.

The simulator acts as a master server for the distributed control clients, communicating with a request-response methodology. It receives Modbus packets requesting to either send simulation data to respective clients or to write control data from them. The software managing this communication is written in C and uses multiplexing of signals (through UNIX socket programming) from different clients to handle all data via the same TCP port (for Modbus communication, port 502 is used). Synchronization with the clients is achieved by calculating and introducing appropriate delays while modeling clients, also discussed in subsequent chapters.

4.2. HIL Client: OrionLX

OrionLX^{TM} is a substation automation platform designed and manufactured by *NovaTech LLC* to perform a range of automation applications in electrical substations. Some of its functions when used in the field include: behaving as a smart remote terminal unit (RTU) for SCADA, as a controller for Intelligent Electronic Devices (IEDs) and as a relay communication processor for protective relays, recorders and monitoring equipment [77].

For our study, the COTS device OrionLX is used as a programmable controller for fault isolation in the simulated microgrid. It uses data received from autorecloser-enabled overcurrent protection relays in the grid. The dynamic grid simulation is done on the real-time platform Opal-RT as described in the previous section. The Ethernet interface setup for OrionLX is done via a Windows-based configuration software *NovaTech Communications Director (NCD).* From the variety of IED protocols supported by OrionLX, Modbus TCP/IP is used for relay data communication to/from the grid.

The logic engine for OrionLX is called *Advanced Math & Logic* and uses the *Lua* programming language [78] for configuring OrionLX control features. Simple logic for fault isolation is written in Lua and programmed into OrionLX. OrionLX regularly polls the simulated relays in the grid for their status and as soon as it detects an overcurrent fault, the relay logic programmed in it is activated and issues control commands to the relays for fault isolation. While this study shows only one case for its use, the HIL methodology can be used to test the physical hardware OrionLX under various simulated conditions.

The *OrionLX Webpage* [77], a web-based HMI specially designed for OrionLX, can be used for viewing data values, port communications and other controller related information when connected through a LAN network. It also allows users to configure its settings and perform diagnostics. OrionLX also logs and archives all events (user initiated and internal) which can be viewed through the OrionLX Web-page.

4.3. Multi-Platform Distributed Clients

As shown in Figure 4.1, aside from the HIL client OrionLX, there are various other distributed clients with their own unique platforms for control algorithms. These are also a part of the client-server architecture, with Opal-RT as the server. The clients are individual processes running on external PCs and communicating with the server through Modbus TCP/IP protocols. The network and software architecture for these clients is explained in detail in the subsequent chapters. The clients are divided into two major categories and are explained as follows:

4.3.1. Feedback Control Clients.

As the name suggests, the feedback control clients use feedback signals from the server to process their programmed algorithms and send control signals back to the server. In our study, two MPPT algorithms are used to control four different PV systems in the simulated grid. The algorithms are executed on two different platforms:

- C# programming language with built-in and custom .NET libraries on *Microsoft Visual Studio* IDE is used to write the P&O MPPT algorithm. The algorithm is used to control two of the four PV system on the server.
- A graphical programming language *Simulink* with *Incremental Conductance* MPPT algorithm written with the help of a control logic tool *Stateflow.* It is used to control the remainder two PV systems on the server.

4.3.2. Data Source Client. The data source client performs unidirectional communication (from client to server only) and is used to send the irradiance profile data values to the four PV systems. A C# code extracts the data values from a time-stamped *Microsoft Excel* spreadsheet on the machine and sends them to the server via a Modbus TCP/IP network using the same libraries written for the feedback control clients. A one-second resolution is used for the irradiance values and hence data is sent once every second.

CHAPTER 5

Software Architecture

The high-level software design of the proposed system, known as the *Software Architecture* [79], has been explained in this chapter. The ideology behind software design decisions and implementation of various abstraction layers for the object-oriented programmed client software is explained. Since synchronized and robust communication is the core of this study, and communication inherently depends on the software-level design, the layered communication network and the protocols used are explained in detail.

The layered communication network architecture takes its basis from the International Standard Organization's *Open System Interconnection (OSI)* reference conceptual model [80] and the *TCP/IP* model [81], as shown in Figure 5.1. The figure shows layered software architecture diagram of the system for both client and server, with the bottom three layers (Physical/Data Link, Network and Transport) representing the network architecture of the system and the rest top-most (Application) layers representing the software architecture. A brief functional description of each layer is also given.

The network architecture layers are explained briefly as follows:

- *Physical/Data Link Layer*: Ethernet networking technology is used because of its widespread and convenient use in Local Area Networks (LANs) and it provides functionality for both Physical and Data Link layers, as referenced to the OSI model. It specifies the physical and electrical specifications of the data connection and ensures reliable point-to-point communication between nodes. The IEEE 802.3 standard Ethernet with 100Base-TX support (known as *Fast Ethernet*) [82] is used in this study.

- *Network Layer*: The fourth revision of the Internet Protocol (IP), IPv4 is used as the network layer. The connection-less protocol, used for packet-switched networks, is responsible for packet routing and network addressing [83]. As shown in Figure 1.1, the distributed networked clients are provided with unique IP addresses to connect to the static-IP server in our study.
- *Transport Layer*: Transmission Control Protocol (TCP) is used as the transport layer, built on the IP protocol. It provides the reliable sending of data packets over a network (error handling, sequencing and acknowledgment) [84]. It can also be integrated easily into the Modbus protocol frame.

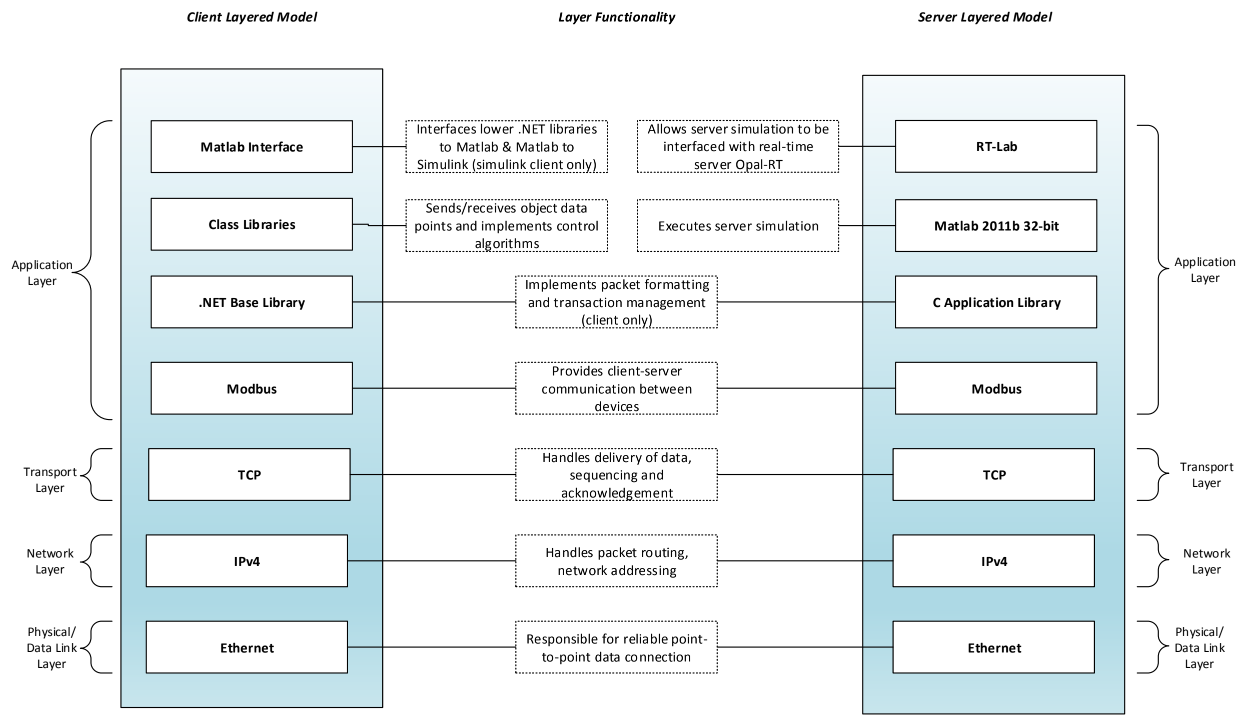

Client Layered Model
Layer Functionality
Server Layered Model
Application Layer
Matlab Interface
Interfaces lower .NET libraries to Matlab & Matlab to Simulink (simulink client only)
Allows server simulation to be interfaced with real-time server Opal-RT
RT-Lab
Class Libraries
Sends/receives object data points and implements control algorithms
Executes server simulation
Matlab 2011b 32-bit
.NET Base Library
Implements packet formatting and transaction management (client only)
C Application Library
Modbus
Provides client-server communication between devices
Modbus
Transport Layer
TCP
Handles delivery of data, sequencing and acknowledgement
TCP
Network Layer
IPv4
Handles packet routing, network addressing
IPv4
Physical/ Data Link Layer
Ethernet
Responsible for reliable point-to-point data connection
Ethernet

Figure 5.1. Layered Network Architecture for Client-Server Model

The highest layer in the OSI architecture, the *Application Layer* is an abstraction layer which consists of communication protocols serving the end-user with information regarding management and servicing of applications [80]. As shown in Figure 5.1, the application layer for client and server differ by the type and number of protocols (and programming layers) used in the application. With respect to the client-server model of Figure 5.1, this layer is explained for both as follows:

- *Server Application Layer*: The application layer on the server side consists of a Modbus protocol layer embedded in the C programming application libraries. Both of these are explained in much detail in subsequent sections of this chapter. These layers facilitate master-slave communication with the clients, with the server behaving as a slave device. Multiple clients are connected to the server which provides seamless communication when clients are added/removed to/from the system.
- *Client Application Layer*: The application layer on the clients is a bit more complicated due to the variable nature of multi-platform clients. Hence the programming layers differ slightly for every type of client to conform to its specifications and architecture. While the Modbus layer is the same as on the server side, it has been implemented in C# using the .NET Base Libraries., provided by *InGreenium LLC* and Mr. Sai Pradeep. The Base Library is capable of handling a number of protocols; Modbus/TCP has been used for our study. Class libraries are built on top of the Base Library which allow for multiple client communications from a single platform. Algorithms for controlling server devices are also implemented there. A Matlab interface is provided to access the lower level libraries for the Simulink platform client interface.

5.1. Modbus Protocol

Modbus is an application layer messaging protocol (Figure 5.1) which allows for client-server communication between several devices connected to the same network. Modbus is especially popular with industrial automation devices and has become the industry's de facto standard. It can be implemented using three types of network architectures [85], namely:

- Ethernet over TCP/IP
- Asynchronous Serial Transmission
- Modbus Plus (a high speed token passing network)

Since with Internet, the prevalence of TCP/IP networks have increased over time for COTS devices connected to control networks, and because of its ease of implementation, Ethernet TCP/IP is used with Modbus for distributed control communication in this study. With the embedding of Modbus frame into a TCP frame, the *Modbus/TCP* is formed [39]. The client-server protocol uses a request-response message passing mechanism to read plant data from the server and write control signals to it. This is shown in Figure 5.2.

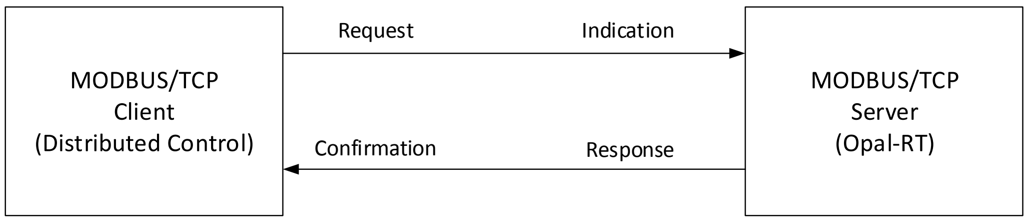


Figure 5.2. Modbus Client-Server Model

The above figure shows the four types of Modbus transaction types, listed below chronologically:

(1) *Request*: Request message packet sent by the client over the TCP/IP network to start an event.

(2) *Indication*: Request message packet received by the server.

(3) *Response*: Response message packet sent by the server back to the client.

(4) *Confirmation*: Response message packet received by the client, completing a single Modbus transaction.

The above methodology is further explained, with an emphasis on distributed control clients, in the next section. Port number 502 is used for Modbus client-server communication.

5.1.1. Modbus Frame. The Modbus request/response frame format over TCP/IP is as shown in Figure 5.3. The full message frame is known as the *Application Data Unit (ADU)*, which contains the *Protocol Data Unit (PDU)* and the *Modbus Application Protocol (MBAP) Header* [86]. All fields are encoded in *Big-Endian* byte ordering notation [87].

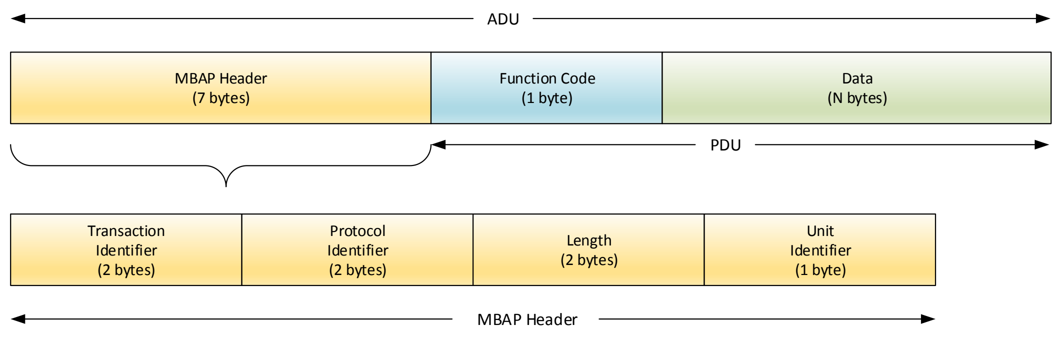


Figure 5.3. Modbus/TCP Frame Format

The seven bytes long MBAP header is added to the start of the messaging frame. It mainly contains ID tags for device and transaction management purposes. The four fields in this header are briefly explained (with their particular usage in this study) as below:

- *Transaction Identifier*: This field identifies the Modbus request/response messaging by keeping track and pairing of the transactions in clients and servers. The ID is assigned by the client initiating request and is recopied by the server in its response packet. In our software architecture, the transaction IDs of multiple clients on the same platform is managed by*Dictionary* type in the .NET Base Libraries.
- *Protocol Identifier*: This field is always zero when implementing Modbus protocol and has been reserved for future protocol extensions. It is also assigned by the initiating client and is recopied by the server in response.
- *Length*: It includes a byte count of all the following fields in the ADU. Since the length of all the other fields in the ADU is known beforehand, this field is especially helpful in determining the number of bytes in the Data field, or in other words, the number of data points. It is assigned separately by both client and the server where they indicate their respective number of following bytes in this field.
- *Unit Identifier*: For a multi-server and/or a multi-client system (like in our study), it is necessary for the initiating client to identify its unique server (unit) on the other end of the communication network. For this purpose, the Unit ID field is used for the unique client-server pairing for information exchange and hence the server should recopy this field while responding to a client request. Since the Unit ID is only 1 byte long, the maximum number of clients that can be connected to a single server, without modifying the protocol format, is 255.

The actual data and the type of request issued by the client (function code) is contained in the PDU. Since it is solely concerned with the data being communicated, the PDU is independent of underlying networking layers. The two fields in PDU, with emphasis on our study, are as explained briefly below [85]:

- *Function Code*: The one-byte field tells the server as to what kind of function to perform with respect to a request from the client (Figure 5.4). The range of function codes is 1-255, where the codes 128-255 are reserved for exception handling. Only a few of the function codes have been used in our application and they are explained as follows:

  (1) *03: Read Holding Registers*: This function code is used by the client to indicate to the server that it wants to read registers from the indicated device. 'Holding Registers' are the registers in which addressing starts from the value 40000 (old PLC terminology). All the distributed multi-platform clients in our application use this function code to read *double* data values from the corresponding simulated device on the real-time server.

  (2) *06: Write Single Register*: This function code is used to write a single holding register in the remote device. In our study, the HIL client *OrionLX* uses this function code to write to simulated IEDs in the grid. The control signals for all IEDs are embedded in this single 16-bit register.

  (3) *16: Write Multiple Registers*: This function code is used to write to a block of contiguous holding registers in the remote device. All the distributed clients (except OrionLX) use this function code to write *double* data values to the server.

- *Data*: The data field of messages sent from a client to server devices contains additional information that the server uses to take the action defined by the function code. This can include items like starting addresses of registers, the quantity of items (data points) to be handled, and the count of actual data bytes in the field. Since in our application, the server *Opal-RT* uses *Simulink* to simulate device objects, all the data points to be read or written have to be in *double* (64-bit float) format. The problem lies in the fact that the Modbus protocol defines one *short* (16-bit float) as one data point. To circumvent this issue, four *shorts* for each *double* data point have been used while reading/writing data points to/from the server. This is achieved programmatically through byte-manipulation while formatting Modbus packets in the server code.

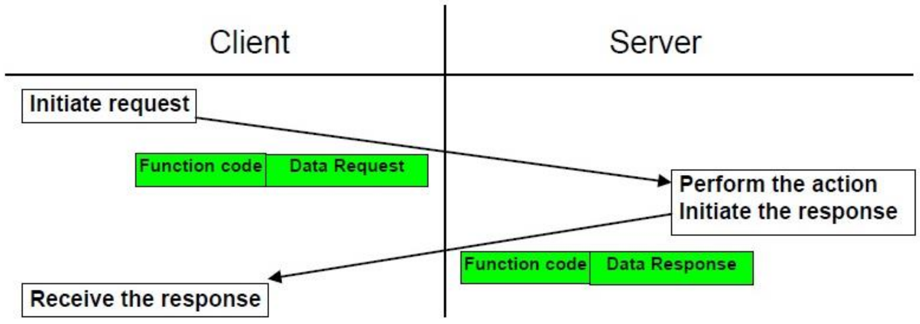


Figure 5.4. Modbus Client-Server Transaction

The choice of Modbus as the application layer protocol in our study has been influenced by various factors like ease of implementation and wide-spread use in automation industry. From subsequent chapters it can also be seen that because Modbus has very small overhead (as compared to more complex protocols like DNP3), response time remains low even for a large number of clients. Real-time synchronization of client-server communication is also achieved easily as seen in the next chapter.

Figure 5.5 shows the complete software architecture for the proposed multi-client, single-server system. The system configuration is as described in Chapter 4, with distributed controls implemented on multi-platform clients connected to a single master server for real-time grid simulation. The client-server communication takes place via Modbus TCP/IP protocols as described in the last section. This and the following sections of the current chapter explains in detail the various software layers constructed and implemented on the master server and individual distributed clients to form the complete software architecture of the system.

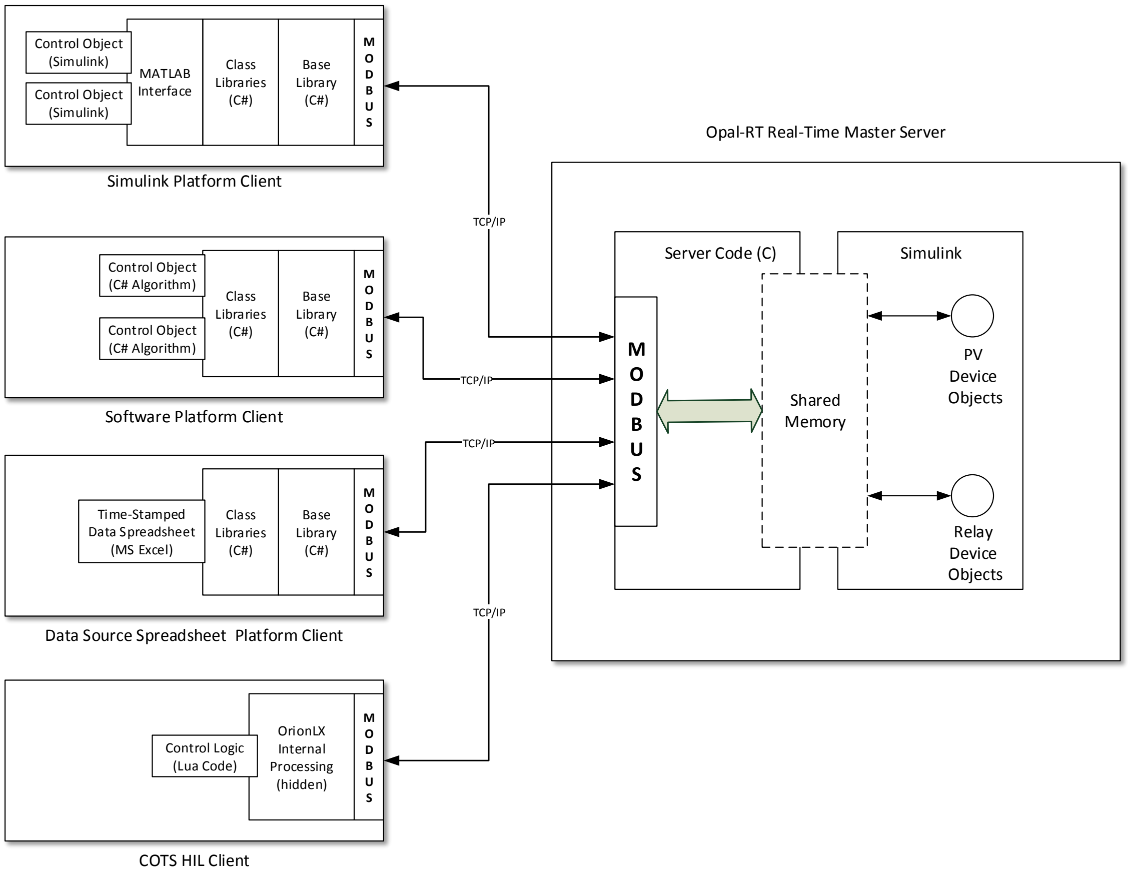

Control Object (Simulink)
Control Object (Simulink)
MATLAB Interface
Class Libraries (C#)
Base Library (C#)
MODBUS
Simulink Platform Client
Control Object (C# Algorithm)
Control Object (C# Algorithm)
Software Platform Client
Time-Stamped Data Spreadsheet (MS Excel)
Data Source Spreadsheet Platform Client
Control Logic (Lua Code)
OrionLX Internal Processing (hidden)
COTS HIL Client
TCP/IP
Opal-RT Real-Time Master Server
Server Code (C)
Simulink
Shared Memory
PV Device Objects
Relay Device Objects

Figure 5.5. Software Architecture for Multi-Client Single-Server Model

5.3. Server Software Development

This section explains in detail the various components and software layers constructed on the server side of the system. A block diagram for the implemented server software architecture is as shown in Figure 5.6. Since the real-time server *Opal-RT* only supports additional software development in C/C++ programming languages, the coding for network programming and Modbus application layer implementation has been done in C. The main components of the block diagram in Figure 5.6 are as follows:

- *Network Socket*: The block provides an interface for client-server communication by implementing socket programming with GNU C libraries for UNIX operating systems (OS) since the server *Opal-RT* uses *RedHat* OS which is UNIX-based.
- *Opal-RT Interface*: The interface provides Simulink blocks and shared memory access for interfacing custom networking libraries with the simulation objects in Simulink. These objects are a part of the system being executed in real-time.
- *Simulated Device Objects*: These are the simulated power system devices which are being controlled via distributed controllers.

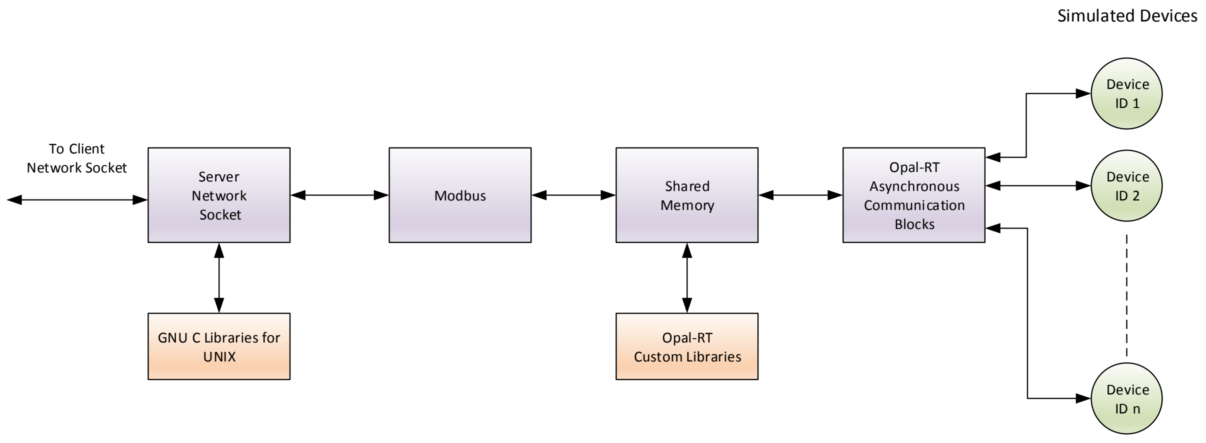


Figure 5.6. Server Software Architecture

5.3.1. Server Network Socket. The two blocks (server network socket and GNU C libraries for UNIX) in Figure 5.6 corresponds to the layers: Physical, Data Link, Network and Transport as were shown in Figure 5.1. The C code for server socket programming has been built from scratch and a state-chart showing event flows for blocking I/O sockets is as shown in Figure 5.7.

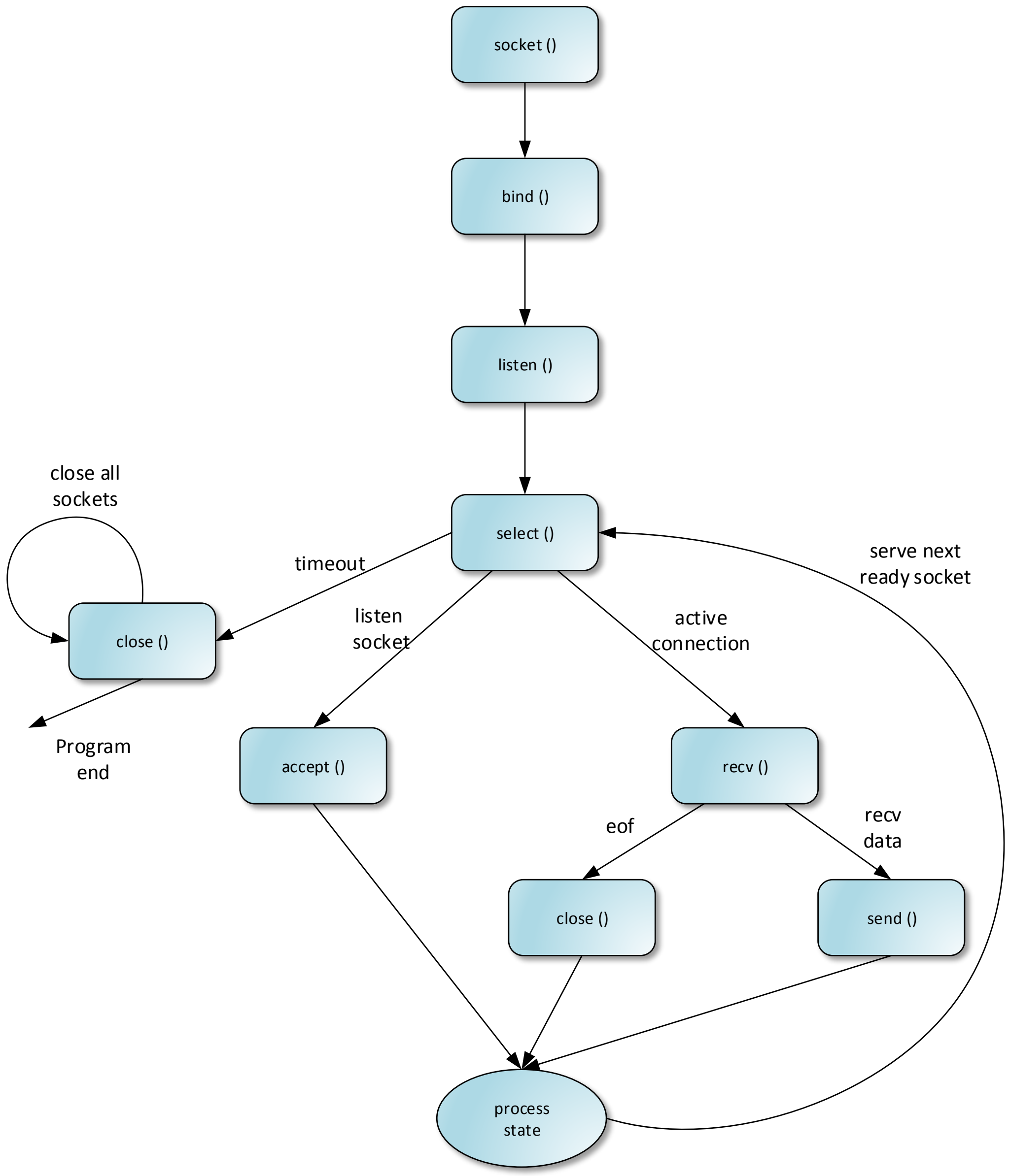


Figure 5.7. Server Socket Flow of Events

The GNU C libraries facilitate inter-process communication on UNIX or UNIX-like OSs using sockets (communication channels). The software as depicted by a flowchart in Figure 5.7 first constructs a socket on the server, waits for incoming client connections and finally sends/receives data as required to/from the specific client and goes back to waiting for more connections. Multiple client-handling capability is also implemented. The numerous API (Application Programming Interface) calls used in Figure 5.7 are described briefly in sequential order as follows:

(1) The **socket()** call returns a socket descriptor which represents an endpoint. Usage of IP and TCP protocols is also specified with this call.

(2) The **bind()** call binds an address and port number to the created socket, on which the server will run (Opal-RT in our case).

(3) The **listen()** call allows the server to accept incoming client connections.

(4) The **select()** call is used for synchronous I/O multiplexing of clients and enables the server to listen for incoming connections as well as read/write data to other clients.

(5) The **accept()** call causes the process to block until a client connects to a server. It wakes up the process when a connection has been established and assigns a new file descriptor for communication with connected client.

(6) The **recv() and send()** calls are used to read and write data from/to the connected client. Non-blocking reads have a timeout of few seconds. After each send() call the process state is ascertained with a call to the shared libraries of Opal-RT. If a model reset has been executed, the close() call is then initiated to end the process.

(7) The **close()** call closes any open socket descriptors.

5.3.2. Opal-RT Interface. As shown in Figure 5.6, the following three blocks constitute the Opal-RT interface:

- *Shared Memory*: The shared memory access for Opal-RT system facilitates the interface between the server simulation and the application server code. It is mainly used to channelize incoming or outgoing data (in *double* precision) to the SimPowerSystems grid simulation from the distributed control clients.
- *Opal-RT Custom Libraries*: The Opal-RT custom libraries are included as header files to the application server code and provide functionality such as facilitating print commands in RT-Lab console for debugging purposes, checking process state and access to the shared memory.
- *Opal-RT Asynchronous Communication Blocks*: An asynchronous process is started by the socket controller block (OplSocketCtrl), where the port number and IP address of the remote host is also provided. As shown in Figure 5.8, the asynchronous communication (Opal-RT exclusive) blocks allow the simulation to send/receive data from the asynchronous process started by the associated socket controller [76].

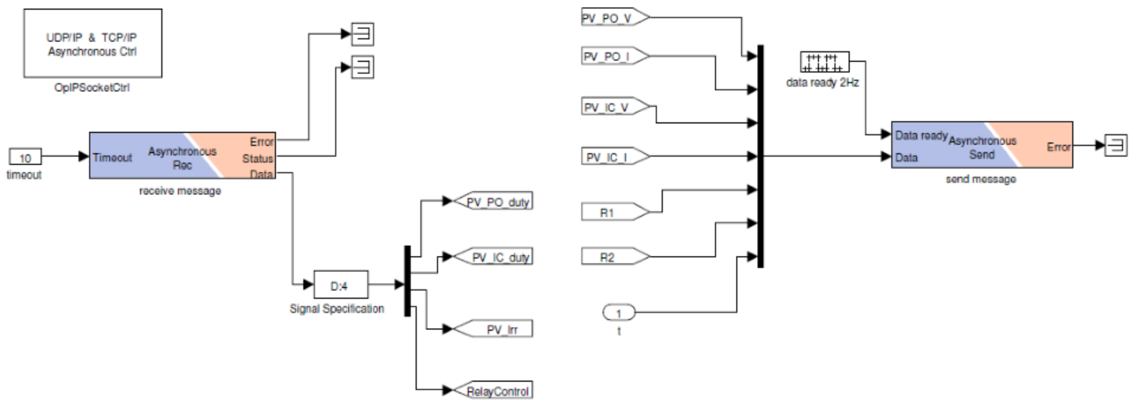


Figure 5.8. Opal-RT Interface

5.3.3. Simulated Device Objects. As explained in Chapter 4, the power output of the four PV systems and overcurrent relay control for fault isolation is being done by the distributed control clients. Also the irradiance profile for the four PV systems is being generated by an external networked data source client. To identify all of these device 'objects' to their respective client controllers, the following two-step procedure is implemented:

(1) **Step 1**: Each server device instance is given a unique ID via the 'unit ID' field of the constructed client Modbus packet. This is done regardless of the nature of the platform on which a client is being executed. In our study, the assigned IDs are as given in Table 5.1.

Table 5.1. Unit ID Assignation for Slave Devices

Server Simulated Device	*Modbus Unit ID*
PV System 1 (P&O)	1
PV System 2 (P&O)	2
PV System 3 (IC)	3
PV System 4 (IC)	4
Irradiance Data Source	5
Relay Control with HIL Client	9

(2) **Step 2**: When initiating communication with a simulated device, a client specifies its device ID by the use of 'unit ID' field in the constructed Modbus packet (see Section 5.1.1 for an overview of Modbus fields). Multiple clients on the same platform are also given unique IDs since they are communicating with different device objects on the server. The server software can then recognize the corresponding device request from the client when multiplexing I/Os.

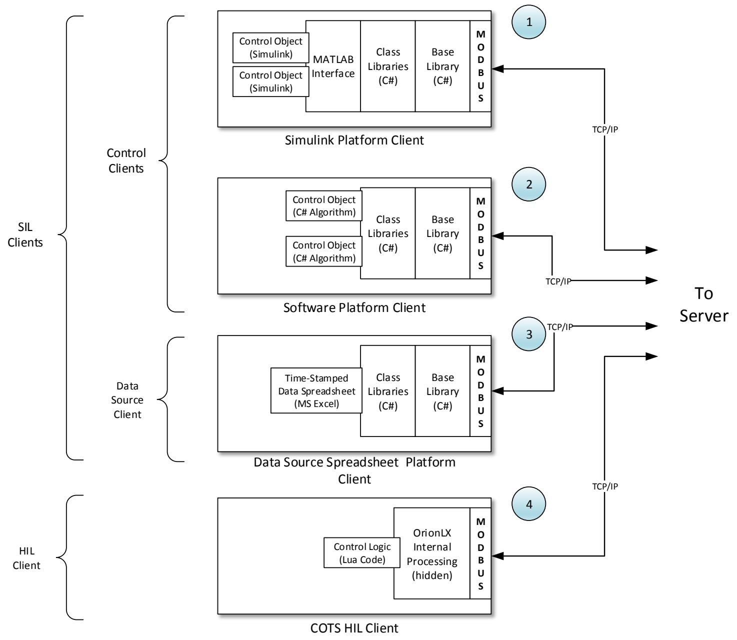
5.4. Client Software Development

The software architecture for various multiple-platform clients, with their classification and numeric labels, is as shown in Figure 5.9. This sections details the multi-layered software development process for each of these four distributed networked clients. The classification of these clients, with regards to their unique properties, is as follows:

(1) *Software-In-Loop (SIL) Clients*: The SIL clients (clients 1-3) are software codes on a networked distributed client node i.e. an external PC. The software is fully written in C# using .NET and custom-made Base Libraries. Many of the same Base Libraries are used in all the clients. They are further divided into:

   (a) *Control Clients*: The clients function as distributed control devices for various simulated devices on the server.

   (b) *Data Source Client*: The client function as an external spreadsheet data source for server devices (no feedback loop).



SIL Clients
Control Clients
Data Source Client
HIL Client
Control Object (Simulink)
Control Object (Simulink)
MATLAB Interface
Class Libraries (C#)
Base Library (C#)
MODBUS
Simulink Platform Client
Control Object (C# Algorithm)
Control Object (C# Algorithm)
Software Platform Client
Time-Stamped Data Spreadsheet (MS Excel)
Data Source Spreadsheet Platform Client
Control Logic (Lua Code)
OrionLX Internal Processing (hidden)
COTS HIL Client
1
2
3
4
TCP/IP
To Server

Figure 5.9. Client Software Architecture with Classification

(2) *Hardware-In-Loop (HIL) Client*: The HIL client (client 4) is a real physical system controlling simulated devices on the server. The control algorithm is being written in *Lua* programming language and internal processing is done by the COTS client for formatting of network packets.

5.4.1. Base Libraries and Class Libraries. The Base Libraries (called *InGreenium.dll* and *TransactionManager.dll*) for client processes are first developed in C# using .NET libraries. These libraries, implemented as abstraction layers in the application, perform the following functions:

- *Marshalling and UnMarshalling*: Marshalling is the process where internally represented data are converted to a format suitable for transmission over a network [88, 89, 90]. The reverse process is called UnMarshalling. This is done so that the remote server can easily interpret the possibly different argument data representations in distributed client's memory. This is a necessary functionality for a flexible client-server architecture since an SIL client can use any one of the large number of application layer protocols available in use for network communication with the master server. In our experimental case study (Chapter 6), only the Modbus/TCP protocol is used for all the clients.
- *Transaction Management*: The transaction management library keeps track of the local network traffic by logging all incoming and outgoing transactions to/from the clients. This is necessary for correct client-server packet deliveries in the case of multi-client objects on the same distributed client platform. It achieves this with the use of .NET *Dictionary* class [91].

- *Endpoint Connections*: A communication endpoint is an interface created and exposed by both the client and server applications to facilitate communication [92]. Its management is also required since a distributed client can use any of the available endpoint connections for communication, for instance network or serial endpoint connections. In our study only network endpoint (with addressing done by port number and IP address) is used for all clients.

The Class Library is more or less the same and common to all the SIL clients. Its architecture, with the use of Base Libraries, is as shown in Figure 5.10. The *Domain Objects* are the SIL client memory representations of data values that are to be communicated through the network to the server. For marshalling, these are first marshalled into Modbus packet formatting and the particular transaction is logged. The values are then converted to a byte stream buffer specific to the protocol being used and communicated over the network via an endpoint connection. For unmarshalling, the reverse process is followed where the received network data bytes are converted to domain objects with unmarshalling. Both marshalling and unmarshalling have read and write capabilities from/to the server.

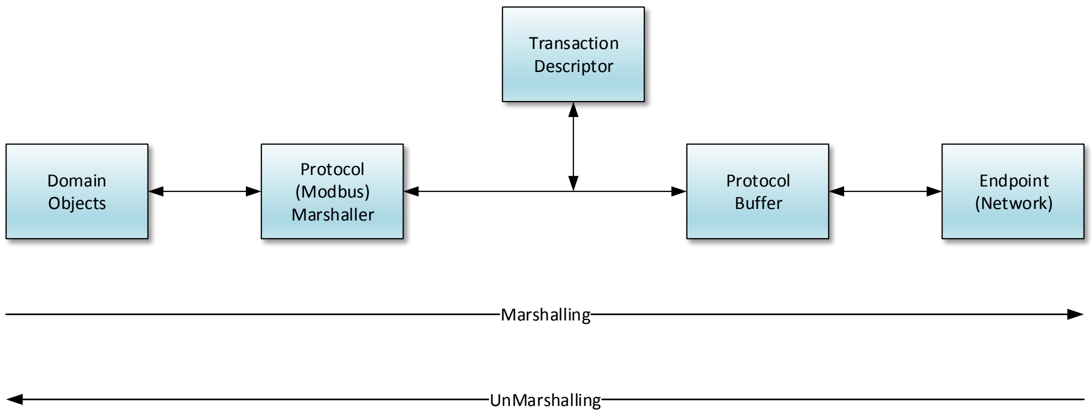


Figure 5.10. Class Library Architecture with Base Libraries

5.4.2. Matlab Interface. For the Simulink platform client (Client No.1 in Figure 5.9), there is an additional Matlab layer after the Class Library for interfacing .NET libraries to Simulink/Matlab environments. The architecture for this (along with various environment extensions) is shown in Figure 5.11.

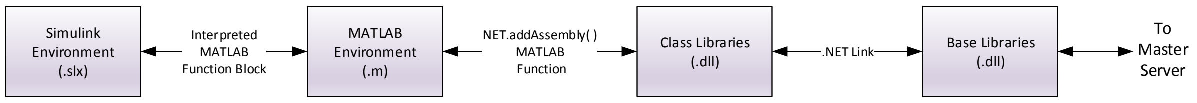

Simulink Environment (.slx)
Interpreted MATLAB Function Block
MATLAB Environment (.m)
NET.addAssembly( ) MATLAB Function
Class Libraries (.dll)
.NET Link
Base Libraries (.dll)
To Master Server

Figure 5.11. Matlab Interface Layer Architecture

The functionality of the added blocks in Figure 5.11 is described below:

- *Simulink Environment*: The control algorithms for controlling server device objects are implemented using Simulink graphical programming language. In our study, the IC MPPT controls of two of the four simulated PV devices is implemented using Stateflow charts in Simulink. Matlab environment access is provided using the *Interpreted Matlab Function Block* from the Simulink library. This block calls a specific Matlab function with Simulink input and output arguments with *double* precision.
- *Matlab Environment*: The Matlab environment then makes pre-compiled custom external .NET assemblies visible to Matlab with the use of *NET.addAssemly( )* method. Classes, methods and properties can be accessed easily from the loaded assemblies using Matlab language. The procedure also loads the .NET *System.dll* library automatically into the Matlab environment. In our case study, it is used to load the base and class libraries developed for client-server communication (See

Section 5.4.1). *Model Callbacks* in Simulink is used to include Class Library methods which are to be used only once and not on every step of the simulation, for instance connection to the server.

5.4.3. User Interfaces. The Base Libraries as well as the Class Library for a particular SIL client have been written as abstraction layers so that the end-user only has to deal with updating domain objects, modifying control algorithms and provide server addressing (port number and IP address of the remote server). The HIL client *OrionLX* uses the *NCD* as its user interface for providing remote server addressing and modification of the control logic built in *Lua* programming language.

CHAPTER 6

Case Study: Multi-Process Microgrid Simulation

A case study involving a simulated microgrid system with distributed networked controls is explained in this chapter. Figure 6.1 shows the same microgrid system constructed in Chapter 3, but now simulated with networked devices such as distributed MPPT control for PV systems, supervisory relay control and irradiance profile external data source. These networked devices function as either SIL or HIL simulations as explained in Chapters 4 and 5. This chapter also discusses the synchronization of these devices with the microgrid simulation, the results of the multi-process simulation and finally the response-time analysis for networked systems.

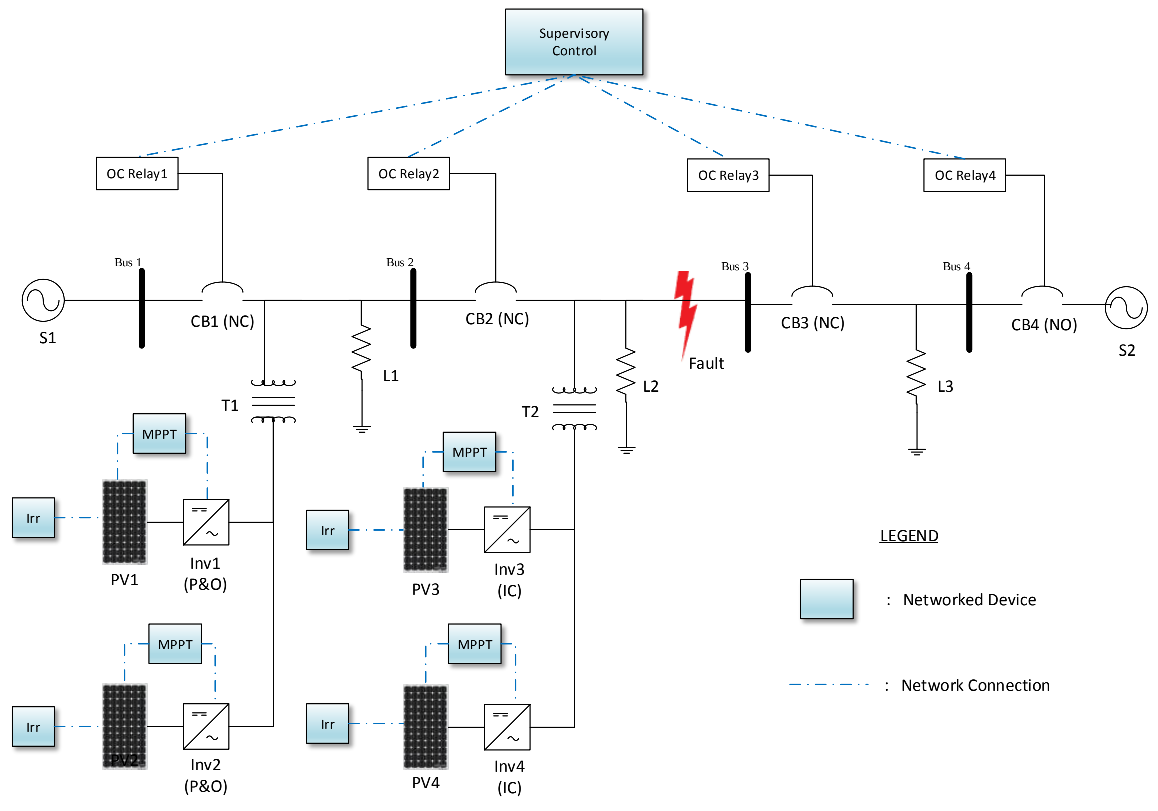


Figure 6.1. Microgrid One-Line Diagram with Distributed Networked Control

6.1. Timing and Synchronization

The case study simulation, when executed on the server *Opal-RT* is found to be approximately 2.6 times slower than real-time. This issue has not been rectified in this study so that the synchronization[1] implementation for the proposed framework can be shown. Hence, since the case-study simulation is not hard real-time, synchronization is needed between the client and server.

Figure 6.2 shows the timing synchronization diagram for the proposed system. The upper horizontal time-axis is for the plant simulation on the server, while the lower time-axis represent the client controller (real) time. Three message transactions are shown in the figure with T_{1-3} being the time-stamp values obtained from the server. The time-stamp values t_{1-6} represent the client time and time-stamps T_{1-3} represent the server time. The first transaction is initiated by the control client at time t_1 which reaches the server at time T_1 and the acknowledgement, with the server time-stamp, is received at time t_2. After executing the necessary processing, the control client will then wait a certain amount of time τ_1 (calculation/control of which is as explained below) before the next client transaction is initiated. It is to be noted that the request period of any client cannot be less than the response time of the same client over the given network. This limitation on client request period is given by:

$$t_{i+1} - t_i < \tau_i \tag{7}$$

[1]It is to be noted that the term 'Synchronization' used in this thesis represents synchronized communication for the client-server architecture and is not related to the synchronization of energy sources with the electric grid, as is used in the power system studies context

where i is the transaction index.

Equation (7) is further explained in Section 6.3. The computation time on the server side is found to be negligible and hence not included in the discussion.

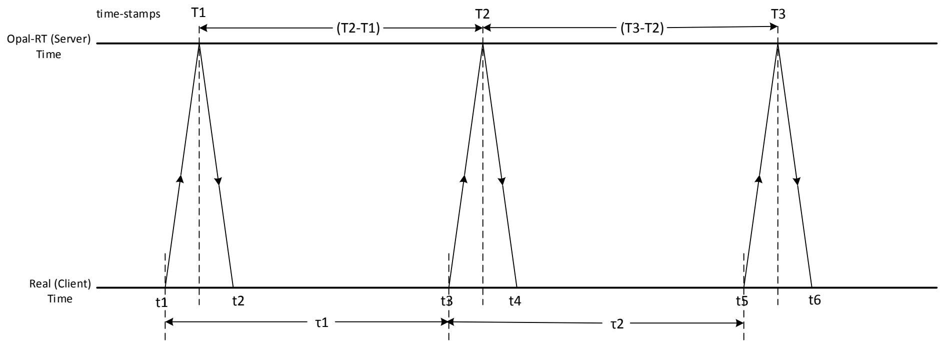


Figure 6.2. Timing Diagram for Client-Server Communication

With reference to the timing diagram in Figure 8, the synchronization objective is given by:

$$T_2 - T_1 = T_3 - T_2 = ... = T_n - T_{n-1} \tag{8}$$

where n is the number of client-server transactions.

The objective in (8) is achieved by controlling the wait-delay τ on the controller simulation. *Bang-Bang Control* strategy [93] is used to synchronize the feedback controller time with the server time. An intelligent guess is made for initial value of τ as $2600 msec$ and it is updated every k transactions to switch to the correct synchronization value. The value of k can be chosen as required by the application and is chosen to be $k = 3$ for this study. This strategy is implemented in all the control clients.

6.2. Simulation Results

The results of the case study for the distributed controlled microgrid simulation are provided in this section. After starting the simulation, the Opal-RT server continuously monitors the port 502 for a client connection request while running the microgrid simulation with default control signal values. Any number of clients can connect/disconnect at any time during the simulation. If a client disconnects while the simulation is running, the simulation object will retain the last control signal it received and continue executing. Control actions of one client can affect results from another. As shown in Figure 6.1, the two main categories in which the distributed control for this case study can be divided are:

- *Fault isolation with Supervisory Control*
- *MPPT Control for PV Systems*

All the control clients on their respective platforms have been executed simultaneously and the results of the simulation are as described:

6.2.1. Fault Isolation with Supervisory Control.

The supervisory control written inside the COTS client *OrionLX* monitors the OC Relays (1-4) and facilitates fault isolation in case of a three-phase overcurrent fault between bus 2 and 3. The control logic written in the supervisory controller is specific to the described fault for the purpose of demonstrating a test case scenario. The series of events occurring during the fault isolation process are outlined below:

(1) An overcurrent three-phase fault occurs at $t = 2secs$ between bus 2 and 3, as shown by the red lightning bolt in Figure 6.1.
(2) Buses 1 and 2 pick up the fault and start their reclosers, as explained and analyzed in Chapter 3, Section 3.3.

(3) Since the *OrionLX* supervisory control is continually polling the relays for fault detection, the status of relays 1 and 2, as provided to the supervisory control, change from 'No Fault' to 'Fault'.

(4) On buses 1 and 2, there is no action taken by the supervisory control and hence relay 2 locks-out, opening CB2 and relay 1 resumes normal operation, closing CB1.

(5) The supervisory control signals relays 3 and 4 so as to open CB3 and close CB4. This control action overrides the autorecloser logic built into the relays.

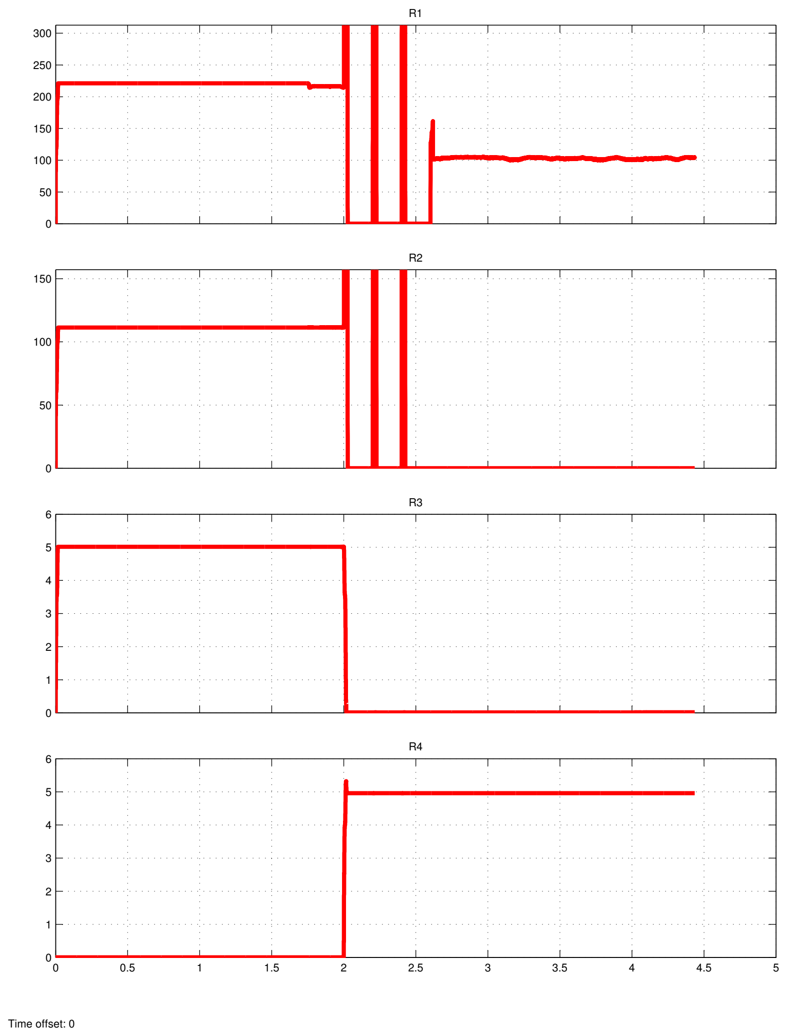


Figure 6.3. Current values for Feeder Fault Isolation with Supervisor Control, showing time increment on the x-axis and current (amperes) on the y-axis

Following the above sequence of actions, the fault is isolated between bus 2 and 3 and hence the loads represented by L3 are now serviced by the back-up source S2. Comparing these results with the simulation carried out in Chapter 3, it is observed that the relay communication through the supervisory control provided by *OrionLX* prevents the loads on L3 from losing service by isolating the fault. It is to be noted that for the purpose of emphasizing communication and not the power system study itself, the supervisory logic built into *OrionLX* is programmed for fault isolation between Bus 2 and Bus 3 only. Automatically detecting a fault and then isolating it is a part of future work of this study. The RMS current analysis diagram of this simulation is shown in Figure 6.3.

6.2.2. MPPT Control for PV Systems. The four PV systems of the microgrid simulation have their MPPT controls as well as the irradiation data source on distributed platforms acting as clients. As described in Chapter 3, the *Perturb & Observe* method is used for PV1 and PV2 while the *Incremental Conductance* method is used for PV3 and PV4. A request period of 1 sec (Opal-RT time) is used by all MPPT control clients for polling of data and sending of control signals. The simulation results for duty cycle analysis (for PV1 and PV3) is as shown in Figure 6.4. Because of slow polling rate of the controllers, the control signal is visible as discrete steps.

The irradiance profile in this case is also not constant and is provided by *National Renewable Energy Laboratory (NREL)* as one-second resolution solar irradiance data for noon, 1st March, 2011 as measured at *Oahu Solar Measurement Grid, Hawaii, US*. The irradiance profile, as seen by the *Opal-RT* server simulation, is as shown in Figure 6.5. The duty cycle control signals are seen to follow the curvature of irradiance profile, verifying its correct functioning.

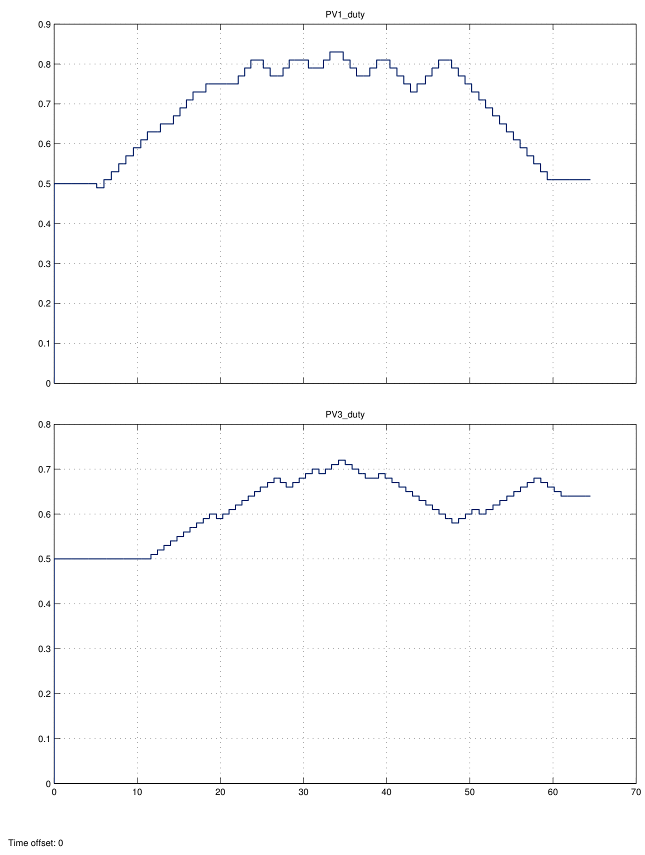


Figure 6.4. Duty Cycles for Networked Control Microgrid PV Systems, showing time increment on the x-axis and duty cycle value on the y-axis

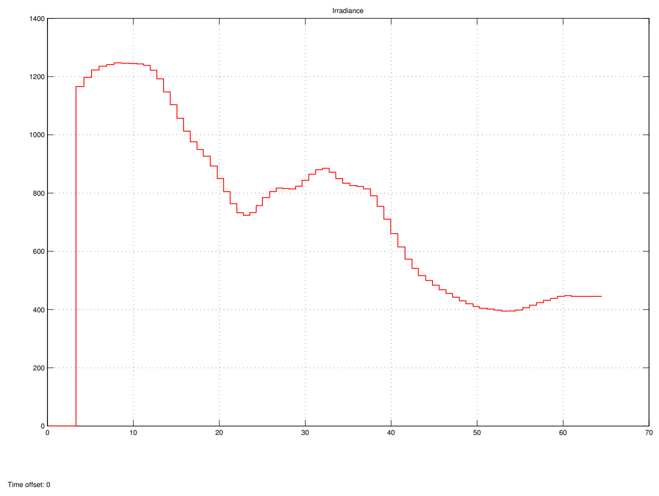


Figure 6.5. Irradiance Profile for PV Systems (Source:NREL), showing time increment on the x-axis and irradiance value (W/m^2) on the y-axis

6.3. Response-Time Analysis

In the context of the client-server model, *Response-Time* is defined as the time difference between the moment (event) the client sends a request out to a server and the moment it gets back an acknowledgment. *Read/Write (R/W) Response-Time* is then defined as the response time of a client when it completes a *read* (acquiring control inputs from the simulation) and *write* (providing control signals to the simulation) cycle. The R/W Response-Time is calculated with the use of *System.Diagnostics.Stopwatch* .NET class, and implemented in C# on the client code. The total number of 'ticks' per R/W cycle are calculated and then divided by the frequency of these timer ticks. This number, when multiplied by $10 \times e^6$ gives the result in microseconds for the amount of time passed. This information for our study is important as it sets a limit on the type of power devices that can be simulated in this 'networked control' configuration.

6.3.1. Response-Time Performance.

For performance analysis in terms of response-times of clients, $10,000$ iterations of R/W polls are done on the server in study and the response-times for each iteration is noted and analyzed. Because all of the clients are executed on the same host PC, it is observed that a small fraction of the data packets give much worse response-times than the others. This is most probably due to the host PC's time-shared execution of multiple processes (each client is executed as a different process, in addition to the default processes executing on the machine). The two types of R/W data packets with respect to their response-times are then defined as:

- *Good data packets*: The data packets with very low R/W response-times in the range of $0 - 4msecs$. They have a high probability of occurrence and represent the response-times when the limitations on the host machine are not considered.

- *Bad data packets*: The data packets with high R/W response-times in the range of $200 - 250 msecs$. They have a very low probability of occurrence (approximately 1:1000 for one additional client) and occur probably due to the time-shared processing nature of the host machine.

Figure 6.6 shows the cumulative distribution function (CDF) plot for 'good data packet' response-times with increasing number of clients. The CDF plots are color-coded with respect to the number of clients executed parallely, as shown in the figure. As seen, the mean response-time for number of clients ranging from 1-5 is between $1.2 - 1.6 msecs$, while the minimum and maximum response-times range from $0.35 - 3.5 msecs$. The CDF curves gives more high-latency packets (decreasing performance) as the number of clients are increased.

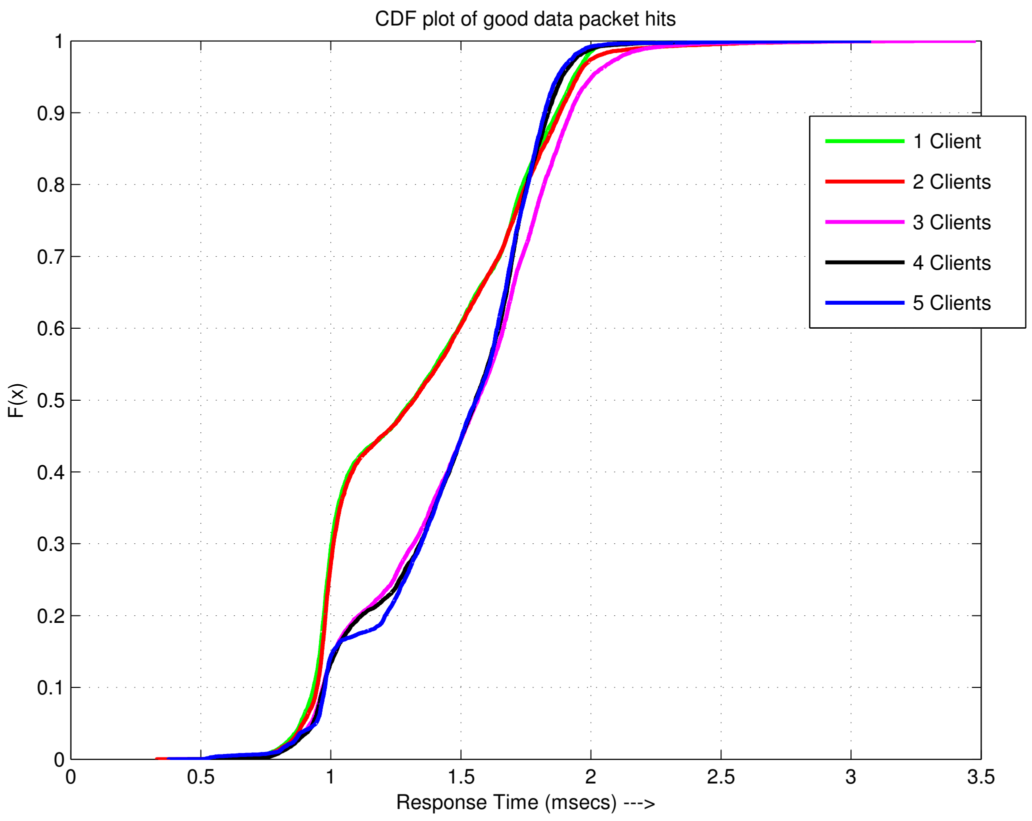


Figure 6.6. CDF Plot for Response-Time Analysis for Increasing Number of Clients)

Figure 6.7 shows the frequency of ‘bad data packet’ hits with increasing number of client connections for 10,000 iterations. As seen from the figure, the frequency occurrence of these packets is low, but it does increase as the number of clients are increased. These data packets are important and can adversely affect the performance of the networked control system by placing a much lower limit on the system-tolerable network delay. Further analysis is required to accurately predict and possibly eliminate the occurrence of these packets.

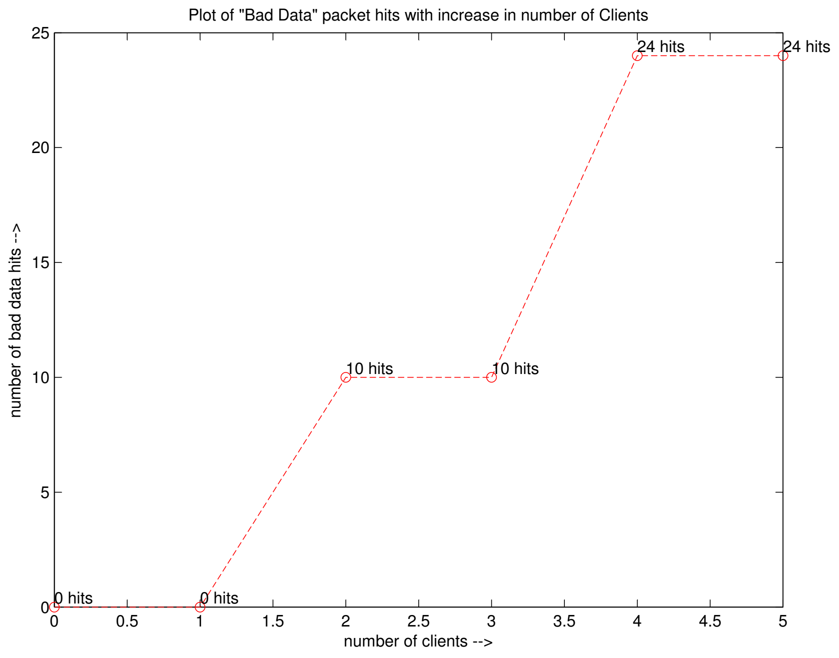
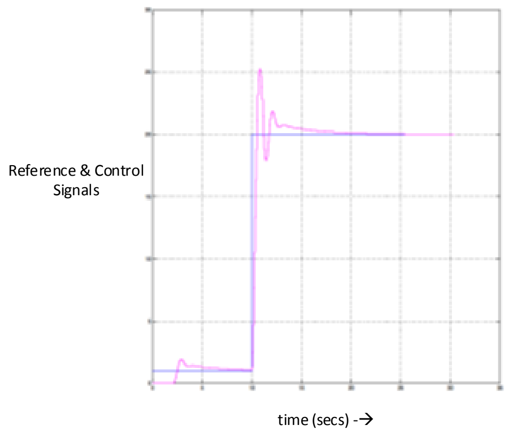
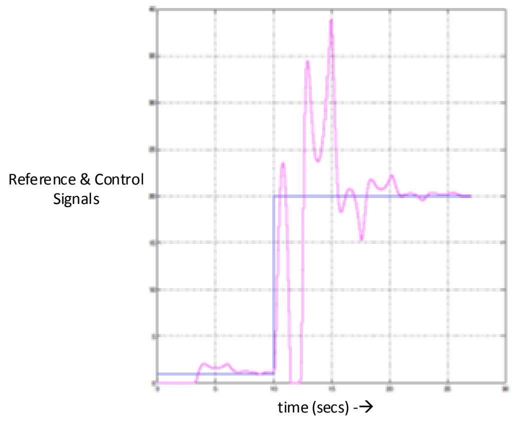


Figure 6.7. Plot showing Number of Bad Data Packets for Increasing Number of Clients

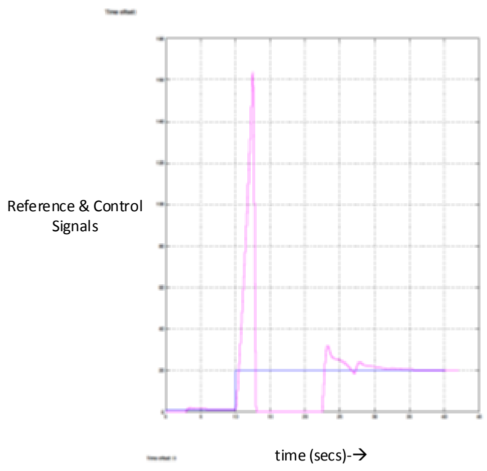
6.3.2. Networked PID Control. A simple PID controller is constructed with the proposed network communication between the reference and the control signals, and its performance for various induced time-delays is analyzed. Figure 6.8 shows the performance degradation of the PID controller step-response with increased forced network delays. The proportional, integral and derivative gains used (without any tuning) for the purpose of this analysis are $K_P = 20$, $K_I = 5$ and $K_D = 5$ respectively.



Forced Network Delay = 0 msecs
Overshoot = 6 pu reference
Settling Time = 15 secs



Forced Network Delay = 5 msecs
Overshoot = 18 pu reference
Settling Time = 17 secs



Forced Network Delay = 10 msecs
Overshoot = 145 pu reference
Settling Time = 30 secs

Figure 6.8. Step-Responses of PID Controller with Various Network Delay Times

CHAPTER 7

Conclusion and Future Work

The multi-disciplinary experimental set-up of this study shows that power system dynamic simulations can be improved by using a combination of parallel and distributed processing. Algebraic control algorithms on various platforms and/or external data sources can be successfully interfaced with real-time simulation in a client-server configuration using Modbus TCP/IP networking protocols. HIL and SIL, both types of simulations can be used to get various simulation advantages such as testing of commercial equipment, less dependency on modeling platforms and modeling environmental characteristics for accurate modeling.

Under the current setup with both 'good' and 'bad' types of data packets potentially affecting the performance and stability of the networked distributed system, the minimum and maximum response times for multiple connected clients are $0.35msecs$ and $250msecs$ respectively. Hence, the distributed networked control framework proposed in this study can be successfully used to network relatively slow algebraic control algorithms, for instance MPPT control for PV systems and supervisory relay control, from the primary simulation. The limitations on networked control will include fast acting dynamic simulations like full-switched bridge inverters, very fast governor controls on synchronous machines etc. There is a trade-off between the number of networked devices connected, their polling frequencies and the performance needed from the distributed system.

With the experimental set-up, it is seen that due to low latency in communication (because of small overhead and frame size), the easily implemented Modbus/TCP protocol can be a good choice for networked controls. It is also a very widely used protocol in automation industry and hence is widely supported by industrial equipment that may need to be tested under the set-up provided in this study.

7.1. Future Work

This study provides a basic framework for *NCS* and *CPS* as related to power systems dynamic HIL & SIL simulations, especially for power system simulations. It can be expanded further in following ways:

- *Synchronization*: A better control mechanism, for instance PID control, can be used for the timing synchronization between client-server communication.
- *Time-Response Analysis*: A more thorough time-response and bandwidth analysis is required for complete understanding of the observed 'bad data packets'. As observed, if these packets can be eliminated from the communication set-up, the performance and list of potential control clients can be significantly improved.
- *Increasing HIL Integration*: The networked controls can be interfaced with the SCADA system supervising a real microgrid, for instance the *InteGrid Lab* at Colorado State University. This will enable the capability to implement control strategies from external software platforms, like Simulink and C# programming languages, and to be tested on real physical systems.
- *NCS Framework Improvement*: Stability and performance improvements can be made in the network architecture of this study by utilizing Networked Control System theoretical studies. The Base Library for networked client devices, provided by *InGreenium LLC*, is flexible enough to incorporate a large number of networking protocols and thus protocols other than Modbus can be used for specific studies.

- *More Complex Control Schemes*: More complex control schemes can be utilized in the current framework, keeping in mind the limitations on network response-times affecting performance and stability.
- *Network Security for CPS*: If incorporated into a complete cyber-physical system architecture, the issue of network security should be addressed.

Bibliography

[1] J. D. Glover, M. S. Sarma, and T. J. Overbye, *Power system analysis and design.* CengageBrain. com, 2011.

[2] D. M. Falcão, "Parallel and distributed processing applications in power system simulation and control," *Revista SBA: Controle & Automação*, vol. 5, pp. 125–143, 1994.

[3] D. J. Tylavsky, A. Bose, F. Alvarado, R. Betancourt, K. Clements, G. Heydt, G. Huang, M. Ilic, M. La Scala, and M. Pai, "Parallel processing in power systems computation," *IEEE Transactions on Power Systems (Institute of Electrical and Electronics Engineers);(United States)*, vol. 7, no. 2, 1992.

[4] D. P. Bertsekas and J. N. Tsitsiklis, "Parallel and distributed computation," *IEEE*, 1989.

[5] X. Wu, H. Figueroa, and A. Monti, "Testing of digital controllers using real-time hardware in the loop simulation," in *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, vol. 5, pp. 3622–3627, IEEE, 2004.

[6] MathWorks, *Simulink User Guide*, r2013b ed., 2013.

[7] G. Karsai and J. Sztipanovits, "Model-integrated development of cyber-physical systems," in *Software Technologies for Embedded and Ubiquitous Systems*, pp. 46–54, Springer, 2008.

[8] A. Umar and C. Fraser, *Distributed computing: a practical synthesis of networks, client-server systems, distributed applications, and open systems.* Prentice-Hall, Inc., 1993.

[9] T. G. Lewis and H. El-Rewini, *Introduction to parallel computing.* Prentice-Hall, Inc., 1992.

[10] B. Stott, O. Alsac, and A. J. Monticelli, "Security analysis and optimization," *Proceedings of the IEEE*, vol. 75, no. 12, pp. 1623–1644, 1987.

[11] N. Balu, T. Bertram, A. Bose, V. Brandwajn, G. Cauley, D. Curtice, A. Fouad, L. Fink, M. G. Lauby, B. F. Wollenberg, *et al.*, "On-line power system security analysis," *Proceedings of the IEEE*, vol. 80, no. 2, pp. 262–282, 1992.

[12] Y. Sekine, K. Takahashi, and T. Sakaguchi, "Real-time simulation of power system dynamics," *International Journal of Electrical Power & Energy Systems*, vol. 16, no. 3, pp. 145–156, 1994.

[13] H. Taoka, I. Iyoda, H. Noguchi, N. Sato, and T. Nakazawa, "Real-time digital simulator for power system analysis on a hypercube computer," *Power Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 1–10, 1992.

[14] D. Brandt, R. Wachal, R. Valiquette, and R. Wierckx, "Closed loop testing of a joint var controller using a digital real-time simulator," *Power Systems, IEEE Transactions on*, vol. 6, no. 3, pp. 1140–1146, 1991.

[15] R. Fujimoto, "Parallel and distributed simulation," in *Simulation Conference Proceedings, 1999 Winter*, vol. 1, pp. 122–131 vol.1, 1999.

[16] P. Venne, J.-N. Paquin, and J. Blanger, "The what, where and why of real-time simulation," in *Power and Energy Society (PES)*, pp. 37–49, 2010.

[17] A. Monti, S. D'Arco, and A. Deshmukh, "A new architecture for low cost power hardware in the loop testing of power electronics equipments," in *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, pp. 2183–2188, IEEE, 2008.

[18] C. Dufour, S. Abourida, and J. Belanger, "Hardware-in-the-loop simulation of power drives with rt-lab," in *Power Electronics and Drives Systems, 2005. PEDS 2005. International Conference on*, vol. 2, pp. 1646–1651, 2005.

[19] J. Wu, Y. Cheng, A. Srivastava, N. Schulz, and H. Ginn, "Hardware in the loop test for power system modeling and simulation," in *Power Systems Conference and Exposition, 2006. PSCE '06. 2006 IEEE PES*, pp. 1892–1897, 2006.

[20] S. Ayasun, S. Vallieu, R. Fischl, and T. Chmielewski, "Electric machinery diagnostic/testing system and power hardware-in-the-loop studies," in *Diagnostics for Electric Machines, Power Electronics and Drives, 2003. SDEMPED 2003. 4th IEEE International Symposium on*, pp. 361–366, IEEE, 2003.

[21] M. Steurer, R. Meeker, K. Schoder, and P. McLaren, "Power hardware-in-the-loop: A value proposition for early stage prototype testing," in *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*, pp. 3731–3735, IEEE, 2011.

[22] W. Ren, M. Steurer, and S. Woodruff, "Accuracy evaluation in power hardware-in-the-loop (phil) simulation center for advanced power systems," in *Proceedings of the 2007 Summer Computer Simulation Conference*, pp. 489–493, Society for Computer Simulation International, 2007.

[23] M. Steurer, F. Bogdan, W. Ren, M. Sloderbeck, and S. Woodruff, "Controller and power hardware-in-loop methods for accelerating renewable energy integration," in *Power Engineering Society General Meeting, 2007. IEEE*, pp. 1–4, IEEE, 2007.

[24] H. D. R. M. López, "Fault location techniques for electrical distribution networks: A literature survey," *IEEE.*

[25] R. Yinger, S. Venkata, V. Centeno, *et al.*, "Fault locating, prediction and protection (flpps)," tech. rep., Southern California Edison Company, Rosemead, CA, 2010.

[26] R. Yinger, S. Venkata, and V. Centeno, "Southern california edison's advanced distribution protection demonstrations," *Smart Grid, IEEE Transactions on*, vol. 3, no. 2, pp. 1012–1019, 2012.

[27] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *Control Systems, IEEE*, vol. 21, no. 1, pp. 84–99, 2001.

[28] Y. Tipsuwan and M.-Y. Chow, "Control methodologies in networked control systems," *Control engineering practice*, vol. 11, no. 10, pp. 1099–1111, 2003.

[29] M.-Y. Chow and Y. Tipsuwan, "Network-based control systems: a tutorial," in *Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE*, vol. 3, pp. 1593–1602, IEEE, 2001.

[30] Z. Yu and M. Yang, "Mfc-based control methodology in network control system," in *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, vol. 2, pp. 1361–1365, IEEE, 2004.

[31] C. Wang and Y. Wang, "Design networked control systems via time-varying delay compensation approach," in *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, vol. 2, pp. 1371–1375, IEEE, 2004.

[32] A. T. Al-Hammouri, M. S. Branicky, V. Liberatore, and S. M. Phillips, "Decentralized and dynamic bandwidth allocation in networked control systems," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pp. 8–pp, IEEE, 2006.

[33] M. Velasco, J. M. Fuertes, C. Lin, P. Marti, and S. Brandt, “A control approach to bandwidth management in networked control systems,” in *Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE*, vol. 3, pp. 2343–2348, IEEE, 2004.

[34] Z. Li and M.-Y. Chow, “Adaptive multiple sampling rate scheduling of real-time networked supervisory control system-part ii,” in *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference on*, pp. 4615–4620, IEEE, 2006.

[35] N. Borisov, I. Goldberg, and D. Wagner, “Intercepting mobile communications: the insecurity of 802.11,” in *Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 180–189, ACM, 2001.

[36] T. Karygiannis and L. Owens, “Wireless network security,” *NIST special publication*, vol. 800, p. 48, 2002.

[37] W.-L. Leung, R. Vanijjirattikhan, Z. Li, L. Xu, T. Richards, B. Ayhan, and M.-Y. Chow, “Intelligent space with time sensitive applications,” in *Advanced Intelligent Mechatronics. Proceedings, 2005 IEEE/ASME International Conference on*, pp. 1413–1418, IEEE, 2005.

[38] A. Casavola and G. Franze, “Coordination strategies for networked control systems: A power system application,” in *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pp. 503–508, IEEE, 2008.

[39] Q. Liu and Y. Li, “Modbus/tcp based network control system for water process in the firepower plant,” in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, vol. 1, pp. 432–435, IEEE, 2006.

[40] C. A. Macana, N. Quijano, and E. Mojica-Nava, “A survey on cyber physical energy systems and their applications on smart grids,” in *Innovative Smart Grid Technologies*

(ISGT Latin America), 2011 IEEE PES Conference on, pp. 1–7, IEEE, 2011.

[41] A. L. Dimeas and N. D. Hatziargyriou, "Operation of a multiagent system for microgrid control," *Power Systems, IEEE Transactions on*, vol. 20, no. 3, pp. 1447–1455, 2005.

[42] N. Cai and J. Mitra, "A decentralized control architecture for a microgrid with power electronic interfaces," in *North American Power Symposium (NAPS), 2010*, pp. 1–8, IEEE, 2010.

[43] S. D. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi, "Multi-agent systems for power engineering applications-part i: concepts, approaches, and technical challenges," *Power Systems, IEEE Transactions on*, vol. 22, no. 4, pp. 1743–1752, 2007.

[44] A. Dimeas and N. Hatziargyriou, "A multi-agent system for microgrids," in *Methods and applications of artificial intelligence*, pp. 447–455, Springer, 2004.

[45] M. Wooldridge, *An introduction to multiagent systems*. Wiley. com, 2008.

[46] J. Ferber, *Multi-agent systems: an introduction to distributed artificial intelligence*, vol. 1. Addison-Wesley Reading, 1999.

[47] T. Hiyama, M. Kawakita, and H. Ono, "Multiagent based wide area stabilization control of power systems using power system stabilizer," in *Power System Technology, 2004. PowerCon 2004. 2004 International Conference on*, vol. 2, pp. 1239–1244, IEEE, 2004.

[48] T. Nagata and H. Sasaki, "A multi-agent approach to power system restoration," *Power Systems, IEEE Transactions on*, vol. 17, no. 2, pp. 457–462, 2002.

[49] J. A. Hossack, J. Menal, S. D. McArthur, and J. R. McDonald, "A multiagent architecture for protection engineering diagnostic assistance," *Power Systems, IEEE Transactions on*, vol. 18, no. 2, pp. 639–647, 2003.

[50] Z. Ming, R. Jianwen, L. Gengyin, and X. Xianghai, "A multi-agent based dispatching operation instructing system in electric power systems," in *Power Engineering Society General Meeting, 2003, IEEE*, vol. 1, pp. 436–440, IEEE, 2003.

[51] C. Y. I. Chung and S. Oh, "Distributed intelligent microgrid control using multi-agent systems," *Engineering*, vol. 5, no. 1B, pp. 1–6, 2013.

[52] S. Rahman, M. Pipattanasomporn, and Y. Teklu, "Intelligent distributed autonomous power systems (idaps)," in *Power Engineering Society General Meeting, 2007. IEEE*, pp. 1–8, IEEE, 2007.

[53] M. Pipattanasomporn, H. Feroze, and S. Rahman, "Multi-agent systems in a distributed smart grid: Design and implementation," in *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*, pp. 1–8, IEEE, 2009.

[54] R. J. Best, D. Morrow, D. M. Laverty, and P. A. Crossley, "Synchrophasor broadcast over internet protocol for distributed generator synchronization," *Power Delivery, IEEE Transactions on*, vol. 25, no. 4, pp. 2835–2841, 2010.

[55] B. Denis, S. Ruel, J.-M. Faure, G. Marsal, and G. Frey, "Measuring the impact of vertical integration on response times in ethernet fieldbuses," in *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, pp. 532–539, IEEE, 2007.

[56] J. Northcote-Green and R. G. Wilson, *Control and automation of electrical power distribution systems.* CRC Press, 2006.

[57] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga, and S. Hariri, "A testbed for analyzing security of scada control systems (tasscs)," in *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*, pp. 1–7, IEEE, 2011.

[58] B. K. Duncan and B. G. Bailey, "Protection, metering, monitoring and control of medium voltage power systems," in *Industrial and Commercial Power Systems, 2003. 2003 IEEE Technical Conference*, pp. 121–128, IEEE, 2003.

[59] P. Neumann, "Communication in industrial automationwhat is going on?," *Control Engineering Practice*, vol. 15, no. 11, pp. 1332–1347, 2007.

[60] M. O. Faruque, M. Sloderbeck, M. Steurer, and V. Dinavahi, "Thermo-electric co-simulation on geographically distributed real-time simulators," in *Power & Energy Society General Meeting, 2009. PES'09. IEEE*, pp. 1–7, IEEE, 2009.

[61] P. Piagi and R. H. Lasseter, "Autonomous control of microgrids," in *Power Engineering Society General Meeting, 2006. IEEE*, pp. 8–pp, IEEE, 2006.

[62] R. Carnieletto, D. I. Brandao, F. A. Farret, M. G. Simoes, and S. Suryanarayanan, "Smart grid initiative," *Industry Applications Magazine, IEEE*, vol. 17, no. 5, pp. 27–35, 2011.

[63] H. S. Rauschenbach, "Solar cell array design handbook-the principles and technology of photovoltaic energy conversion," 1980.

[64] SunPower Corporation, *SunPower 305 Solar Panel*, 2009.

[65] M. Godoy Simoes, N. Franceschetti, and J. Adamowski, "Drive system control and energy management of a solar powered electric vehicle," in *Applied Power Electronics Conference and Exposition, 1998. APEC'98. Conference Proceedings 1998., Thirteenth Annual*, vol. 1, pp. 49–55, IEEE, 1998.

[66] D. Hohm and M. Ropp, "Comparative study of maximum power point tracking algorithms using an experimental, programmable, maximum power point tracking test bed," in *Photovoltaic Specialists Conference, 2000. Conference Record of the Twenty-Eighth*

IEEE, pp. 1699–1702, IEEE, 2000.

[67] J. J. Nedumgatt, K. Jayakrishnan, S. Umashankar, D. Vijayakumar, and D. Kothari, "Perturb and observe mppt algorithm for solar pv systems-modeling and simulation," in *India Conference (INDICON), 2011 Annual IEEE*, pp. 1–6, IEEE, 2011.

[68] N. Instruments, "Maximum power point tracking," tech. rep., National Instruments, 2009.

[69] A. Safari and S. Mekhilef, "Simulation and hardware implementation of incremental conductance mppt with direct control method using cuk converter," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 4, pp. 1154–1161, 2011.

[70] S. Hong, "Harmonics and noise in photovoltaic (pv) inverter and the mitigation strategies," tech. rep., Solectria Renewables, 2010.

[71] M. Geidl, *Protection of power systems with distributed generation: state of the art.* ETH, Eidgenössische Technische Hochschule Zürich, EEH Power Systems Laboratory, 2005.

[72] G. Electricals, "Distribution system feeder overcurrent protection," 2010.

[73] P. P. Bedekar, S. R. Bhide, and V. S. Kale, "Optimum coordination of overcurrent relays in distribution system using genetic algorithm," in *Power Systems, 2009. ICPS'09. International Conference on*, pp. 1–6, IEEE, 2009.

[74] M. Bacic, "On hardware-in-the-loop simulation," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pp. 3194–3198, IEEE, 2005.

[75] J. Bélanger, L. A. Snider, J.-N. Paquin, C. Pirolli, and W. Li, "A modern and open real-time digital simulator of contemporary power systems," in *Proceedings of the International Conference on Power Systems Transients (IPST 2009), Kyoto, Japan*, pp. 2–6,

2009.

[76] Opal-RT Technologies Inc., *RT-LAB User Guide*, ver.10.4 ed., 2007.

[77] NovaTech LLC, *OrionLX User Manual*, revision j ed., 2013.

[78] R. Ierusalimschy, L. H. De Figueiredo, and W. Celes Filho, "Lua-an extensible extension language," *Softw., Pract. Exper.*, vol. 26, no. 6, pp. 635–652, 1996.

[79] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, pp. 40–52, 1992.

[80] H. Zimmermann, "Osi reference model–the iso model of architecture for open systems interconnection," *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425–432, 1980.

[81] B. A. Forouzan, *TCP/IP protocol suite.* McGraw-Hill, Inc., 2002.

[82] "Ieee standards for local and metropolitan area networks: Supplement to carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications media access control (mac) parameters, physical layer, medium attachment units, and repeater for 100 mb/s operation, type 100base-t (clauses 21-30)," *IEEE Std 802.3u-1995 (Supplement to ISO/IEC 8802-3: 1993; ANSI/IEEE Std 802.3, 1993 Edition)*, pp. 1–398, 1995.

[83] J. Postel, "Internet protocol," 1981.

[84] J. Postel, "Transmission control protocol," 1981.

[85] I. Modbus, "Modbus application protocol specification v1. 1b3," *North Grafton, Massachusetts (www. modbus. org/specs. php)*, 2012.

[86] I. Modbus, "Modbus messaging on tcp/ip implementation guide v1. 0b," *North Grafton, Massachusetts (www. modbus. org/specs. php)*, 2006.

[87] L. Null and J. Lobur, *The essentials of computer organization and architecture.* Jones & Bartlett Publishers, 2010.

[88] S. Kundu, *Fundamentals of Computer Networks.* PHI Learning Pvt. Ltd., 2005.

[89] K. P. Birman, *Reliable distributed systems: technologies, web services, and applications.* Springer, 2005.

[90] P. Hoschka and C. Huitema, "Automatic generation of optimized code for marshalling routines.," in *ULPAA*, pp. 135–150, 1994.

[91] P. Sestoft and H. I. Hensen, *C [Precisely.* The MIT Press, 2004.

[92] J. C. Foster, *Sockets, Shellcode, Porting, and Coding: Reverse Engineering Exploits and Tool Coding for Security Professionals: Reverse Engineering Exploits and Tool Coding for Security Professionals.* Syngress, 2005.

[93] R. Bellman, I. Glicksberg, and O. Gross, "On the'bang-bang'control problem," tech. rep., DTIC Document, 1955.

APPENDIX A

MPPT Algorithms & PV Array Specifications

The flowcharts for the two MPPT algorithms implemented in this study are given in Figure A.1 and A.2.

A.1. Perturb & Observe Method

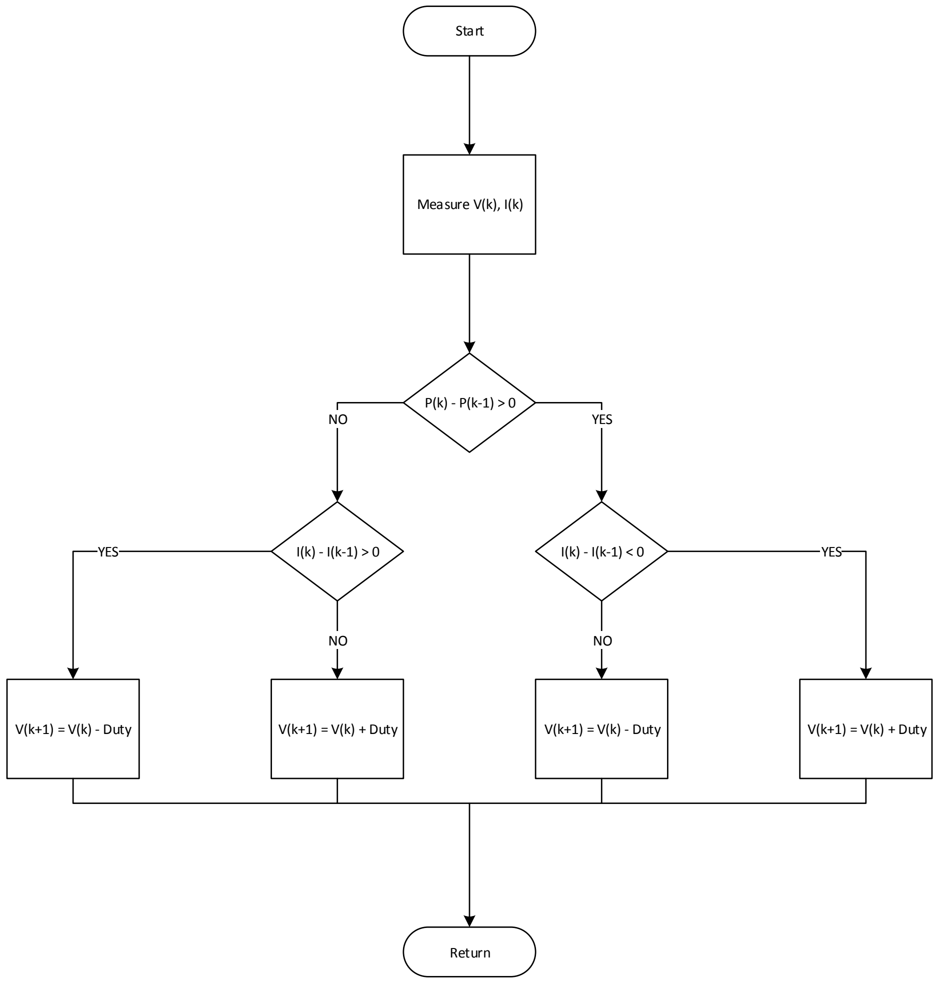


Figure A.1. Flowchart for Perturb & Observe Algorithm

A.2. Incremental Conductance Method

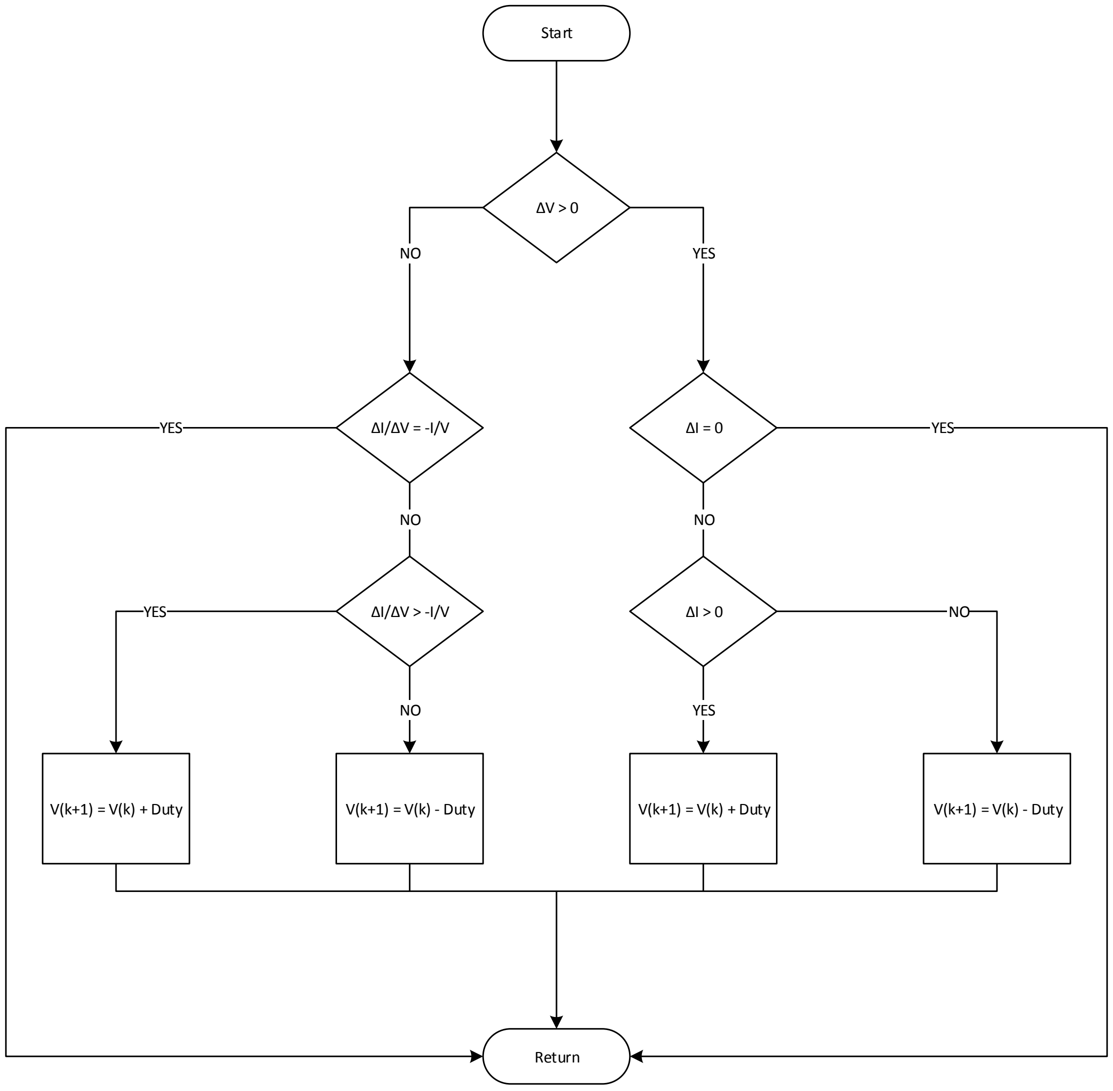


Figure A.2. Flowchart for Incremental Conductance Algorithm

A.3. PV Array Specifications

PV Array specifications for the *SunPower SPR-305-WHT PV Panel* [64] are given in Table A.1.

Table A.1. SunPower SPR-305-WHT PV Array Specifications

Parameter	*Variable (Units)*	*Value*
Number of cells in series	N_{cell}	96
Number of series connected modules per string	N_{ser}	5
Number of parallel strings	N_{par}	66
Maximum Power	$P_{mp}(W)$	305.2
Maximum Power Voltage	$V_{mp}(V)$	54.70
Maximum Power Current	$I_{mp}(A)$	5.58
Open-circuit Voltage	$V_{oc}(V)$	64.20
Short-circuit Current	$I_{sc}(A)$	5.96
Series Resistance	$R_s(\Omega)$	0.0380
Parallel Resistance	$R_p(\Omega)$	993.5
Diode Saturation Current	$I_{sat}(A)$	$3.1949e^{-8}$
Light Generated Photo-Current	$I_{ph}(A)$	5.9602
Diode Quality Factor	Q_d	1.3

Maximum power for a single PV array (Watts) is thus given by:

$$P_{array} = N_{par} \times N_{ser} \times P_{mp} \tag{9}$$

From Equation (9), $P_{array} = 100.7$ kW.