

© 2018 Samuel Utomi

HARDWARE-IN-THE-LOOP PLATFORM FOR THE COORDINATION
OF DISTRIBUTED ENERGY RESOURCES TO PROVIDE
FREQUENCY REGULATION SERVICES

BY

SAMUEL UTOMI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Associate Professor Alejandro D. Domínguez-García

ABSTRACT

With the proliferation of distributed energy resources (DERs) in the power grid, new operational challenges have emerged. By coordinating these DERs effectively, one can provide services to mitigate these effects and provide value to the grid. One such service is frequency regulation, which is among the ancillary services provided in the open market management of the grid, and which has high value as it is important to maintain stability in the grid. This thesis presents a hardware-in-the-loop platform for testing the effectiveness of different coordination schemes for providing frequency regulation services. While coordination of these DERs is usually done in a centralized manner, in our testbed we aim to show the effectiveness of distributed control algorithms, which are used along with local controls to coordinate these DERs and achieve the frequency regulation objective. We first elaborate on frequency regulation and the provision process, then present an overview of our hardware testbed. We then proceed to present the various parts that make up this testbed on both the hardware and software level, describing in detail the distributed algorithm used and how it is developed on our software platform. Finally we present various experiments that showcase the use of our hardware testbed and the results that follow.

To my family and friends, for their continued love and support.

ACKNOWLEDGMENTS

I would like to thank my advisor, Alejandro D. Domínguez-García, for the guidance, dedication and support he provided throughout my master's work. I am grateful for the opportunity to work on this research project and have learned much since I first started. I am also thankful to all the professors and staff in the Power and Energy Systems group at ECE UIUC who not only helped me through my journey but also created a vibrant atmosphere which allowed me to develop and grow as a person.

I would like to express my gratitude to my colleagues Olaolu Ajala, Hanchen Xu, Madi Zholbaryssov, Dariush Fooladivanda, Siddhartha Nigam and all the other students in the Power and Energy Systems group for the great time and wonderful moments spent together laughing and learning, which have made my master's experience memorable.

I am also grateful for the support received from the Advanced Research Project Agency-Energy (ARPA-E) who funded the project and made all of this possible. And I would also like to thank Typhoon HIL Inc. for the equipment and support they provided through the course of the project.

Finally I would like to thank the ECE Department here at UIUC for creating an environment where people with different perspectives can come together to achieve their unique goals and strive for greatness.

TABLE OF CONTENTS

LIST OF TABLES	vi
LIST OF FIGURES	vii
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 FREQUENCY REGULATION	3
CHAPTER 3 SYSTEM CONFIGURATION	6
CHAPTER 4 PHYSICAL LAYER	8
CHAPTER 5 CYBER LAYER	11
5.1 Model	11
5.2 Hardware	12
5.3 Cyber Layer Software	13
CHAPTER 6 EXPERIMENTAL RESULTS	23
6.1 Resource Allocation for 4-Node Case	23
6.2 Resource Allocation for 6-Node Case	26
6.3 Resource Allocation to Minimize System Losses	29
CHAPTER 7 CONCLUSION	34
REFERENCES	36

LIST OF TABLES

5.1	Modbus Register Types	21
5.2	Function Codes	21

LIST OF FIGURES

2.1	System Load With and Without Regulation [5]	4
2.2	Frequency Regulation Provision Process	5
3.1	System Overview	7
3.2	Hardware Testbed	7
4.1	Typhoon HIL 402	8
4.2	Power Controller	9
4.3	Typhoon HIL Modbus Communication Interface	9
4.4	Typhoon HIL Modbus Configuration	10
5.1	Communication Network Graph	12
5.2	Hardware Controller	13
5.3	Cyber Layer Flow Diagram	14
5.4	Random Communication Graph	16
5.5	Evolution of γ for All 10 Nodes	16
5.6	Ratio-Consensus Process per Iteration	17
5.7	Synchronization Process for Two-Node Case	20
5.8	Communication Architecture	20
5.9	Application Data Unit	21
5.10	Client/Server Communication	22
6.1	4 Node Communication Graph	23
6.2	Evolution of γ for 4 nodes (Hardware Testbed)	24
6.3	Evolution of γ for 4 Nodes (Simulation)	25
6.4	RegD Signal vs. DER Power Output	25
6.5	6-Node Communication Graph	26
6.6	Evolution of γ for 6 nodes (Hardware Testbed)	27
6.7	Node 1 Reference Tracking	28
6.8	Node 2 Reference Tracking	28
6.9	Node 3 Reference Tracking	28
6.10	Node 4 Reference Tracking	28
6.11	Node 5 Reference Tracking	29
6.12	Node 6 Reference Tracking	29
6.13	Evolution of Ratio-Consensus when Node 1 Computes μ^*	32

6.14	Evolution of Ratio-Consensus when Computing α	33
------	--	----

CHAPTER 1

INTRODUCTION

Over the past few years, electric power networks have undergone a stark transformation due to environmental concerns and the desire to improve reliability and sustainability. Proliferation of distributed energy resources (DERs) have played a major role and have led to additional complexity in the bulk grid which in turn has led to embedding intelligence in the power network in order to tackle the new challenges. One such challenge is the frequency fluctuation caused by random changes in demand as well as the presence of DERs such as photovoltaic (PV) units, wind turbines, fuel cells, and energy storage elements.

By effectively integrating and coordinating DERs, ancillary services can be provided which can help tackle some of these challenges. The successful coordination of DERs can be realized through microgrids, which are small-scale power systems composed of DERs. These microgrids are autonomous systems with dedicated control systems that guarantee good power quality, where the primary sources of energy of these DER units are interfaced to the microgrid either through conventional means like rotating machines or through power electronic converters. This system could either be DC, AC or hybrid DC/AC, but in our work we focus on an AC microgrid system with inverter-based sources. AC microgrids are common since AC distribution systems are the standard choice for commercial power systems and most loads operate with AC power supply [1]. Such microgrids operate either in islanded or grid-connected mode based on the desired objective. When in islanded mode, the microgrid is disconnected from the bulk grid so the frequency and voltage need to be controlled by the DER interfaces in order to maintain microgrid stability while power is actively controlled in order to meet the load requirements in the system. In grid-connected mode, the microgrid behaves as a single controllable generator connected to the larger electrical system through the point of common coupling (PCC).

In order for the microgrid to perform effectively, it requires a sophisticated control architecture in order to maintain stability and provide necessary services promptly. Unlike conventional power systems, the structure of a microgrid is unique in that it allows the implementation of a distributed control architecture where the decision making is distributed among the DER controllers [2]. This means that rather than relying on a central processor to compute control signals for the DERs, control is achieved locally by exchanging information among neighboring controllers, which increases reliability since there is not a single point of failure.

In order to test the performance of this architecture, a Hardware-in-the-Loop (HIL) Simulator [3] can be used to replace the physical electric AC system composed of DERs which are connected to actual controller hardware with wireless capabilities that allow the implementation of the distributed control algorithm. In a HIL simulation, a real-time computer acts as a virtual representation of a plant model (microgrid) which can interface with the actual controller hardware through various communication interfaces such as analog/digital I/Os or Ethernet. The power and control hardware-in-the-loop simulation systems are two HIL applications used to develop and test complex real-time embedded systems [4].

The power HIL setup is used when the controller is implemented in the simulator and connected via I/O interfaces to the actual controlled object while in the control HIL setup, the virtual controlled object is implemented in the real-time simulator which is connected to an actual controller. The control HIL setup is usually used when focusing on the effectiveness of control algorithms in practice and is the focus of this thesis.

In this work, we choose to use a control HIL setup as it is a great way to test the effectiveness of our control algorithms in early stages of prototyping. With this setup, our control algorithms can be rapidly tested without risking inverter failure due to coding errors, and parameters can be fine-tuned to optimize performance.

CHAPTER 2

FREQUENCY REGULATION

As mentioned in Chapter 1, one of the challenges faced in today's power systems is frequency fluctuations which can lead to grid instability. In order to synchronize generation assets for electrical grid operation and maintain stability, the alternating frequency must be held within tight bounds which can be achieved by providing frequency regulation services. Figure 2.1 shows the difference in the load with and without regulation and emphasizes the need to provide frequency regulation services. This frequency regulation is mainly provided by the ramping up/down of a set of generation assets. This is the case due to the strong relationship between active power and frequency of the electric network which is a result of how synchronous generators, which make up a large percentage of power generation, react to changes in demand. Thus, when frequency fluctuations occur due to random changes in demand, generation assets known as regulating units correct for the unintended fluctuations by adjusting their power output. This helps to maintain frequency, manage differences between actual and scheduled power flows between control areas, and match generation to load within the control area. Control areas do not require perfect balance between generation and demand but must satisfy the permissible imbalance limits established by NERC. These regulating units take part in the real-time regulation market with the PJM Regulation Market being one such example.

PJM Interconnection is a regional transmission organization (RTO) in the United States that coordinates the movement of wholesale electricity in all or parts of 13 states and the District of Columbia. It operates an ancillary services market with regulation being one such service. In its real-time regulation market, it provides two different regulation signals:

- RegD: Fast, dynamic signal that requires regulation units to respond almost instantaneously; and

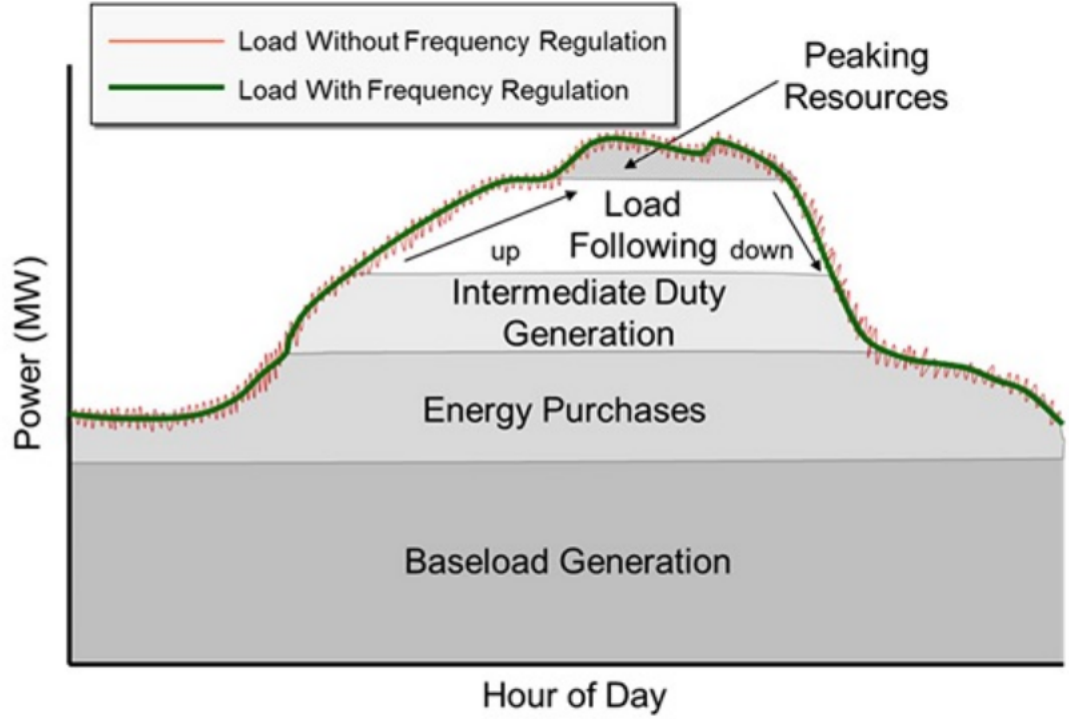


Figure 2.1: System Load With and Without Regulation [5]

- RegA: Slower signal that is meant to recover larger, longer fluctuations in system conditions.

We focus on the RegD signal because it is better suited to microgrid applications which are composed of DERs, such as energy storage, units that can respond almost instantaneously.

The RegD signal corresponds to a required change in power output (with respect to some baseline power) of the microgrid application at the point of common coupling (PCC) and changes every 2-4 seconds. The main goal of this signal is to keep the system's area control error (ACE) within acceptable bounds, where ACE is the difference between scheduled and actual electrical generation accounting for variations in system frequency. Since individual DERs may not meet the size or performance requirement needed to participate in frequency regulation provision, a group of DERs is aggregated to satisfy the requirements. So in the case of a grid-connected microgrid application which is composed of individual DERs, the requirements are aptly met. Specifically, an aggregator is used to act as an intermediary between the microgrid and the system operator or RTO, sending an offer bid for providing

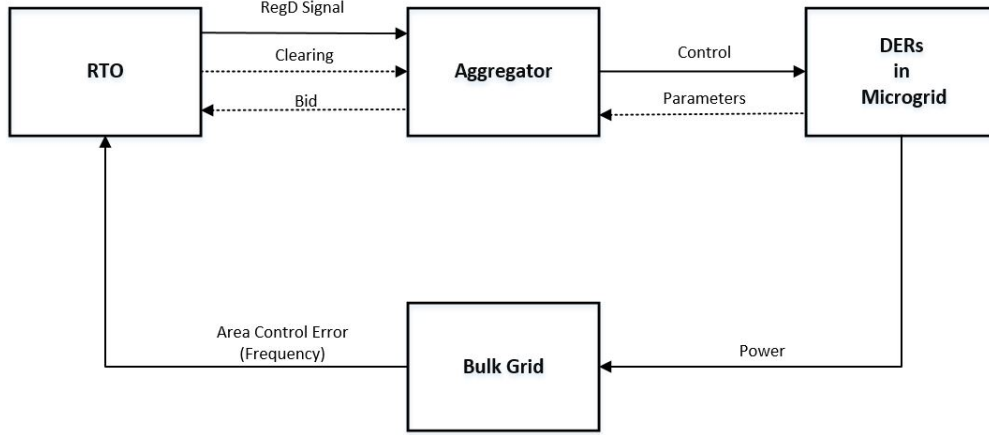


Figure 2.2: Frequency Regulation Provision Process

frequency regulation services to the RTO based on the microgrid’s regulation capacity, and receiving a clearing price as well as regulation signals over a 5 minute period from the RTO; the microgrid must then determine the setpoints for the DERs in order to follow the regulation signal. The regulation signal data is normalized using the maximum regulation capacity of the microgrid, if the RegD signal is positive, and it is normalized using the minimum regulation capacity of the microgrid, if the RegD signal is negative. Figure 2.2 better illustrates the frequency regulation provision process. We implement this process by creating a hardware-in-the-loop testbed that allows us to show the coordination of DERs in a microgrid in response to regulation signals received by the aggregator.

CHAPTER 3

SYSTEM CONFIGURATION

Our hardware-in-the-loop (HIL) testbed is composed of two layers: the physical layer and the cyber layer. The physical layer is represented by a microgrid modeled in the Typhoon HIL simulator [3]. The DERs in the microgrid are DC sources interfaced by inverter modules, and electrically connected to each other via distribution lines. This microgrid is connected to an infinite bus (stiff voltage source), which acts as the bulk grid (i.e., we assume the microgrid operates in grid-connected mode). The cyber layer is composed of hardware controllers, which are assigned to a unique DER, that communicate wirelessly based on the associated communication graph of the cyber layer. Along with this, there is also a computing platform which acts as the aggregator and sends regulation signals obtained from the RTO. This setup is better illustrated in Figure 3.1. In our setup, we ignore the interaction between the RTO and aggregator in real time and instead use pre-determined regulation signal data over a certain period of time obtained from the PJM Interconnection for our experiments. The regulation signal in this case corresponds to a fraction of the microgrid’s regulation capacity, which needs to be provided to the bulk grid, and lies between 0 and 1 ($0 \leq \text{RegD} \leq 1$).

The frequency regulation provision process in our experiments takes place as follows: The computing platform, acting as the aggregator, sends regulation signals to the hardware controller acting as the leader. The other hardware controllers in the network act as followers that wait on the leader to begin the coordination process. Upon receiving this signal, the leader initiates a distributed control algorithm referred to as ratio-consensus [6] (discussed in detail in Chapter 5.3) by first synchronizing with the other controllers in the cyber network via the algorithm proposed in [7]. Synchronization is necessary due to the requirements of ratio consensus and the algorithm to achieve this will also be discussed in more detail later. Once synchronization is complete, the ratio-consensus process is executed over a

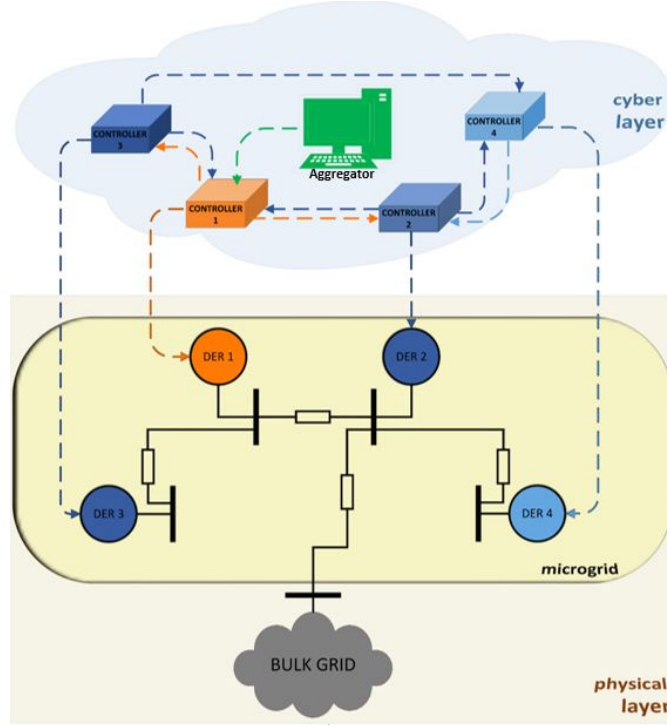


Figure 3.1: System Overview

certain period of time based on the set parameters, and is used to determine how much regulation power each DER provides. The results from the ratio-consensus process are then sent by the hardware controllers to their assigned DER in the microgrid model via Ethernet communication. Each DER in the model has a local power controller which uses the power reference value to determine the setpoints of the inverter modules and ultimately adjust their power output in order to meet the desired regulation objective. The testbed used in our experiments is shown in Figure 3.2.

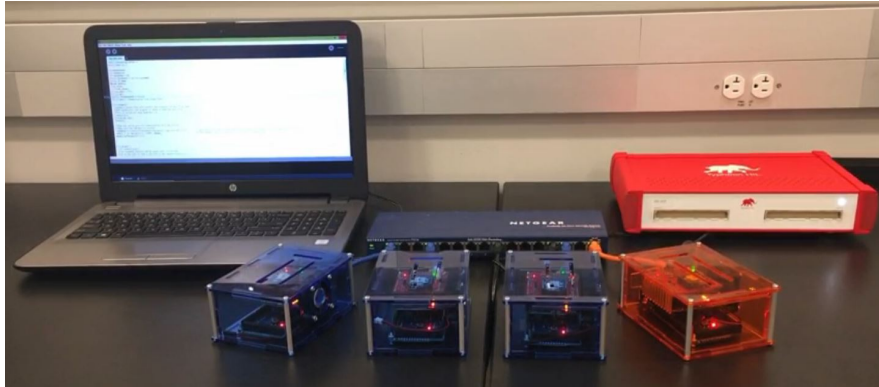


Figure 3.2: Hardware Testbed

CHAPTER 4

PHYSICAL LAYER

For research purposes, rather than go through the time and cost-intensive process of building a microgrid, one can model it in a virtual environment. In our case, we use the Typhoon HIL 402 hardware-in-the-loop system [8], shown in Figure 4.1, as the virtual environment where we create our microgrid model. The HIL 402 is a compact, extremely powerful system with four core processors and is able to communicate with external devices via its analog/digital I/O pins, USB or Ethernet interface.

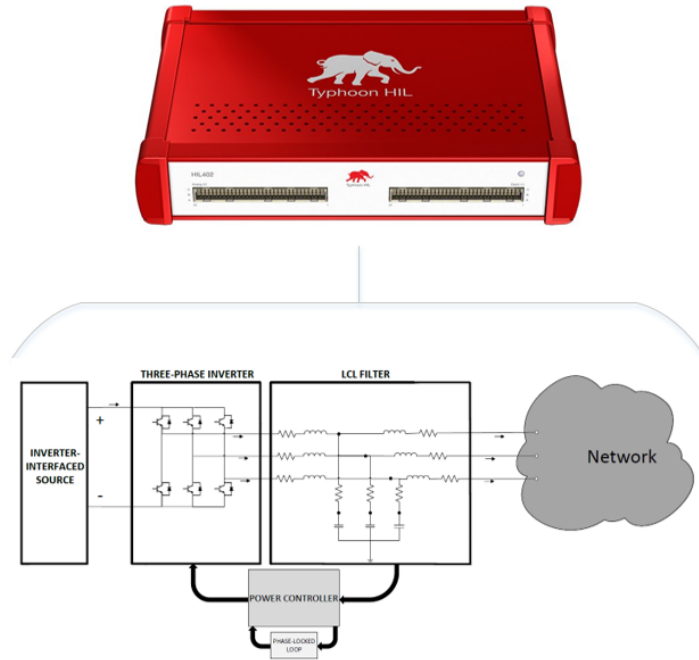


Figure 4.1: Typhoon HIL 402

Each DER in the microgrid is composed of a constant DC source, three-phase inverter, LCL filter, and power controller with phase locked loop. The three-phase inverter converts the voltage and current from DC to AC via the on and off time sequences of the transistors, which are controlled through

a modulation technique, such as pulse width modulation (PWM) [9]. The LCL filter is present to get rid of undesired switching frequency components. The power controller, the block diagram of which is shown in Figure 4.2, consists of an outer power loop and inner current feedback loop structure which tracks the real and reactive power references [10]. The inner current feedback loop maintains the inductor current value and is performed in the direct-quadrature-zero (DQZ) reference frame. This relies on the outer power control loop to provide the inductor current reference value, which is calculated by using the real or reactive power references as well as the output voltage in the DQZ reference frame. The phase locked loop provides the reference angle for the DQZ reference frame.

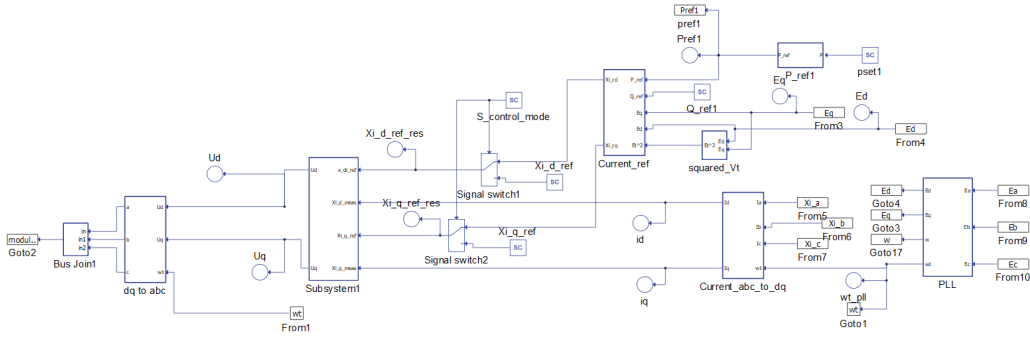


Figure 4.2: Power Controller

Each DER has its own local power controller (emulated in the Typhoon HIL), and receives its power reference signal via a modbus device configured such that it receives information from a dedicated hardware controller in the cyber layer.

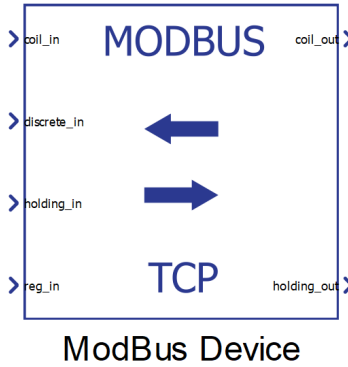


Figure 4.3: Typhoon HIL Modbus Communication Interface

This modbus device as shown in Figure 4.3 is provided as one of the components in the Typhoon library, and is an effective means of communication with our external hardware controllers. An example configuration of this modbus device is shown in Figure 4.4. This is based on the modbus TCP protocol discussed in detail in Chapter 5.

```
config1 = {  
    'port': 502,  
    'ip_addr': '192.168.2.41',  
    'netmask': '255.255.255.0',  
    'slave_id': 1,  
    'coil_input_addresses': '',  
    'coil_output_addresses': '',  
    'discrete_input_addresses': '',  
    'holding_register_input_addresses': '2,3',  
    'holding_register_output_addresses': '0,1',  
    'input_register_addresses': '4,5'  
}
```

Figure 4.4: Typhoon HIL Modbus Configuration

CHAPTER 5

CYBER LAYER

5.1 Model

To show the effectiveness of our distributed control algorithm in a practical setting, we establish a cyber network of hardware controllers each with wireless capabilities that allow each DER controller to exchange information with other DER controllers in the network. Each hardware controller is assigned to a dedicated DER created in the Typhoon HIL system (physical layer).

The communication network interconnecting these hardware controllers is defined by a direct graph $G = \{V, E\}$, where $V := 1, 2, \dots, n$ is the vertex set with each vertex or node corresponding to a DER, and where $E \subseteq V * V$ is the edge set, with the ordered pair $(i, j) \in E$ if node i can receive information from node j . Since this is a directed graph, an out-neighborhood and in-neighborhood of node i are defined; out-neighborhood of node i , $\mathcal{N}_i^+ := \{j \in V : (j, i) \in E\}$, refers to the nodes that can receive information from node i while in-neighborhood of node i , $\mathcal{N}_i^- := \{j \in V : (i, j) \in E\}$, refers to the nodes that can send information to node i . Note that this also includes self-loops; thus node i is a member of its out-neighborhood as well as its in-neighborhood, i.e. $i \in \mathcal{N}_i^+$ and $i \in \mathcal{N}_i^-$, $\forall i \in V$. Furthermore, the cardinality of the out-neighborhood of node i , referred to as the out-degree, is denoted as $D_i^+ := |\mathcal{N}_i^+|$. The out-degree is important because it determines how node i distributes information among its neighbors in the network. The ring graph model of a four-node network with bidirectional communication links is shown in Figure 5.1 as an example. In this work, we assume the directed graph is strongly connected; i.e., for any node i and j in V , there is a path from i to j .

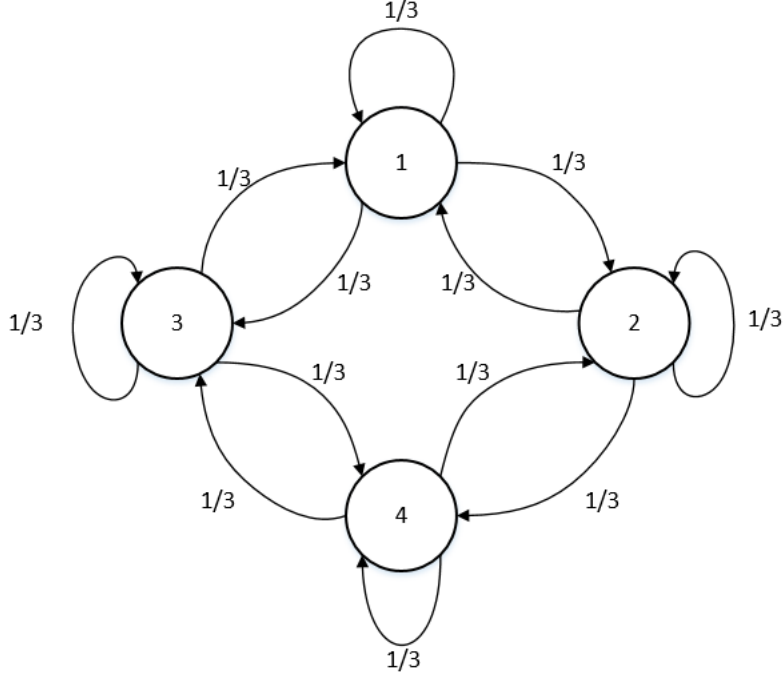


Figure 5.1: Communication Network Graph

5.2 Hardware

Each node in the communication network is comprised of an Arduino Mega 2560 microcontroller board, W5100 Ethernet shield, and a MaxStream XB24-DMCIT-250 revB XBee module via a SparkFun Electronics XBee shield as shown in Figure 5.2. The Arduino Mega is a microcontroller board based on the ATmega 2560. It has 54 digital I/O pins, 16 analog inputs, 4 universal asynchronous receiver transmitter (UART) ports, a 16 MHz crystal oscillator, a USB connection, and a power jack. The W5100 Ethernet shield is connected to the microcontroller board using long wire tap headers, which extend through the shield, and communicates with the Arduino using the SPI bus. The shield has an embedded Wiznet W5100 Ethernet chip which provides a network (IP) stack capable of both TCP and UDP. This allows connection to the internet via Ethernet, but in the case of our application, it is used to establish a TCP connection with the Typhoon HIL system. It has a connection speed of 10/100 MB/s. The XBee shield is stacked on top of the Ethernet shield and provides 3.3 V power supply required by the XBee module. The XBee is an embedded RF module with a built-in chip antenna operating at 2.4 GHz that allows for wireless communication using the Zigbee

protocol. Each XBee has a unique fixed address that distinguishes it from the others in the network, and to communicate wirelessly they must have the same personal area network (PAN) address and operate on the same channel; this can be configured over a set range.

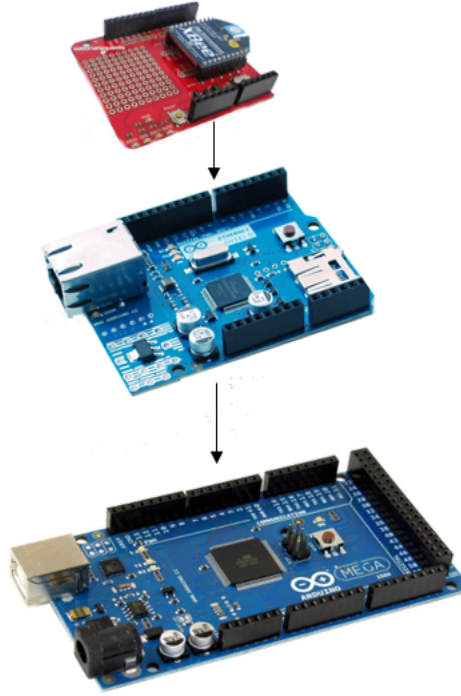


Figure 5.2: Hardware Controller

5.3 Cyber Layer Software

The hardware controllers rely on developed algorithms and protocols that are essential for the testbed to operate effectively. These are: a communication protocol that allows data transfer between the aggregator and the hardware controller acting as the leader, the ratio-consensus algorithm that is used to distribute the resource requested by the aggregator among all the nodes in the network, clock synchronization protocol where all nodes are synchronized to a common time reference, and a modbus TCP protocol that allows data transfer between controllers and the Typhoon HIL system. Figure 5.3 shows a flow diagram highlighting how these protocols and algorithms are

used together to achieve our experiments objective. Next, we discuss these algorithms and protocols in more detail.

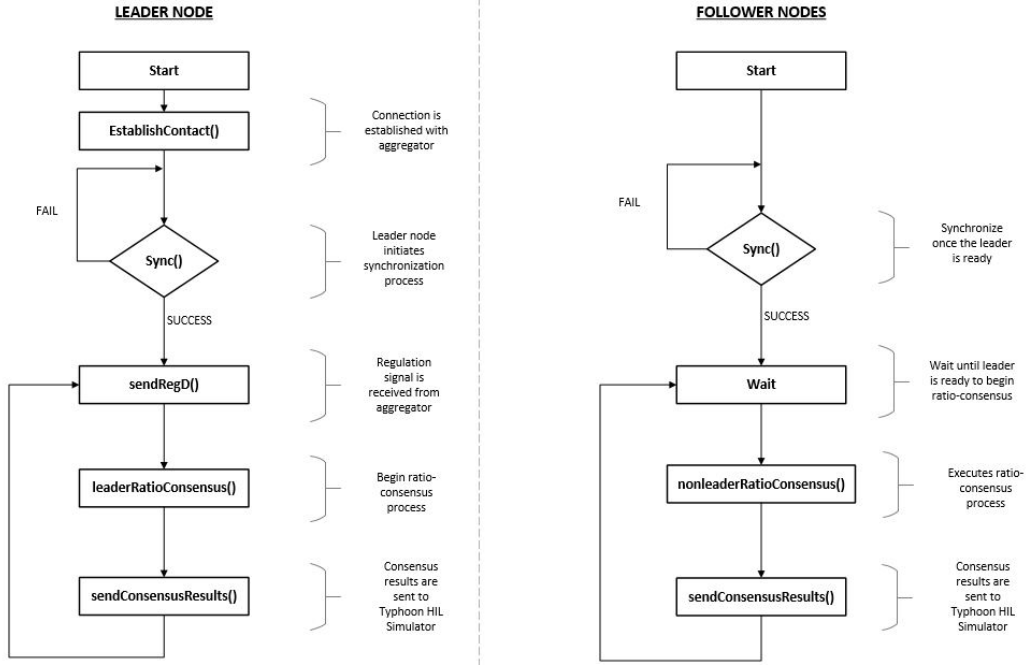


Figure 5.3: Cyber Layer Flow Diagram

5.3.1 USB Communication Protocol

The computing platform, which acts as the aggregator, makes use of Processing [11], a flexible software sketchbook, to send regulation signals to the hardware controller acting as the leader via USB communication. In order for this to take place, the leader first sends a command signal to establish a connection with the aggregator, after which an acknowledgement signal is sent in response. Once this process is complete, the aggregator remains on standby until the leader node sends a data request command which prompts the retrieval of a datapoint from a dataset of regulation signals obtained from PJM. This datapoint is sent through a USB port at a data rate of 38400 bits per second. This process is repeated throughout the duration of the experiment until the dataset of regulation signals is empty.

5.3.2 Ratio-Consensus Algorithm

We use a distributed algorithm known as ratio-consensus (see, e.g. [6]) which relies on two linear iterations executed in parallel by the local controller of each DER appropriately initialized based on the requirements of the problem. In the case of this work, it is used to achieve power sharing between DERs in the microgrid proportional to their capacities in order to provide frequency regulation services to the power grid.

The local controller of each DER i participating in the ratio-consensus process has two internal states, y_i and z_i . Each state is independently updated per iteration to a linear combination of its previous state and the previous states of DERs in its in-neighborhood as follows:

$$\begin{aligned} y_i[k+1] &= \sum_{j \in \mathcal{N}_i^-} \frac{1}{D_j^+} y_j[k], \\ z_i[k+1] &= \sum_{j \in \mathcal{N}_i^-} \frac{1}{D_j^+} z_j[k]. \end{aligned} \quad (5.1)$$

where D_j^+ is the out degree of DER j that lies in the in-neighborhood of DER i . Assuming $z_i[k] > 0, \forall i \in \mathcal{X}$, node i computes

$$\gamma_i[k] = \frac{y_i[k]}{z_i[k]}. \quad (5.2)$$

As follows from [6], for sufficiently large k , all local controllers will converge to

$$\gamma^* = \frac{\sum_{j=1}^N y_j[0]}{\sum_{j=1}^N z_j[0]}. \quad (5.3)$$

To show the effectiveness of the ratio consensus algorithm, we present an example in a MATLAB simulation setting for a 10-node (DER) case with a random communication graph as shown in Figure 5.4.

Based on the aforementioned communication graph, there is an associated weight matrix based on the out-degree of each node in the graph (self-loops are included) which determines how much information is distributed to each node's neighbors. The initial values of y and z are set to $y[0] = [0.8, 0.7, 0.5, 0.4, 0.35, 0.25, 0.45, 0.55, 0.65, 0.85]$ and $z[0] = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$ respectively. As shown in Figure 5.5, all nodes converge to the solution, $\gamma = \frac{\sum_{j=1}^{10} y_j[0]}{\sum_{j=1}^{10} z_j[0]} = 0.55$, in less than 50 iterations.

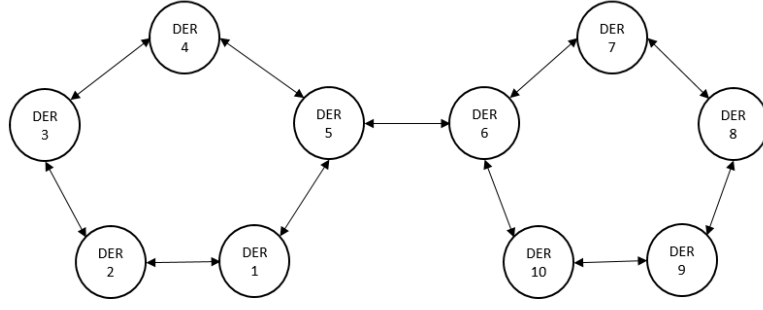


Figure 5.4: Random Communication Graph

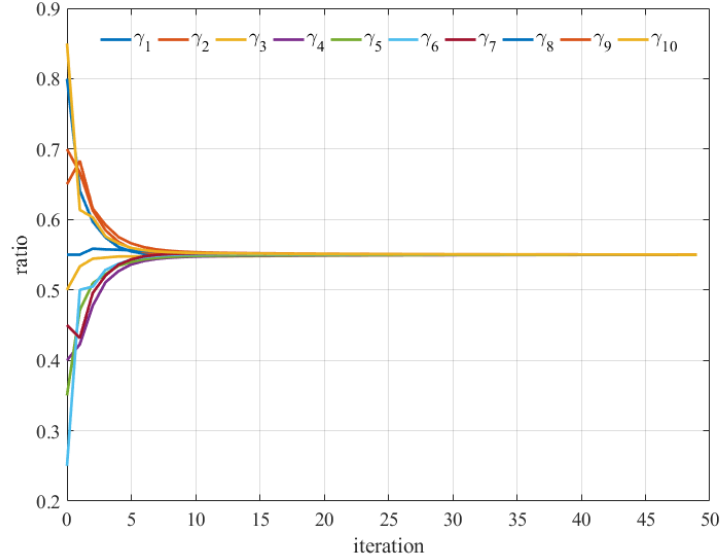


Figure 5.5: Evolution of γ for All 10 Nodes

5.3.3 Robust Ratio-Consensus Algorithm

While the ratio-consensus algorithm works as expected in simulation tests, that is not the case when it is applied in a distributed communication network as is the case with our application. The issue lies in network packet drops inherent in these networks which affect the convergence of the ratio-consensus algorithm. Accordingly, we use a modified version of ratio-consensus [12] whereby each node instead sends the sum of the weighted sums up to and including the current iteration. This modification takes advantage of the iterative process by accumulating information as time progresses such that

even if packet drops occur at one iteration, information is preserved and can be delivered at the next successful iteration, as follows:

$$\begin{aligned}\mu_i[k+1] &= \mu_i[k] + \frac{1}{D_i^+} y_i[k] = \sum_{t=0}^k \frac{1}{D_i^+} y_i[t], \\ \sigma_i[k+1] &= \sigma_i[k] + \frac{1}{D_i^+} z_i[k] = \sum_{t=0}^k \frac{1}{D_i^+} z_i[t].\end{aligned}\tag{5.4}$$

States are then updated as follows:

$$\begin{aligned}y_i[k+1] &= \frac{1}{D_i^+} y_i[k] + \sum_{j \in \mathcal{N}_i^-, i \neq j} (v_{ij}[k+1] - v_{ij}[k]), \\ z_i[k+1] &= \frac{1}{D_i^+} z_i[k] + \sum_{j \in \mathcal{N}_i^-, i \neq j} (\tau_{ij}[k+1] - \tau_{ij}[k]) \\ v_{ij}[k+1] &= \begin{cases} \mu_j[k+1], & \text{(packet received)} \\ v_{ij}[k], & \text{(packet not received)} \end{cases} \\ \tau_{ij}[k+1] &= \begin{cases} \sigma_j[k+1], & \text{(packet received)} \\ \tau_{ij}[k], & \text{(packet not received).} \end{cases}\end{aligned}\tag{5.5}$$

In our software implementation of this algorithm, each node executes ratio-consensus over a certain number of iterations m , with each iteration lasting for a certain period of time Δt . The iteration period Δt is chosen such that the probability of packet collisions resulting from nodes broadcasting their packets concurrently is reduced while the iteration count m is chosen such that convergence to the same γ is possible. At the start, only the leader node is aware of m and Δt . These parameters are distributed to the follower nodes along with the start time t_s for the iterative process.

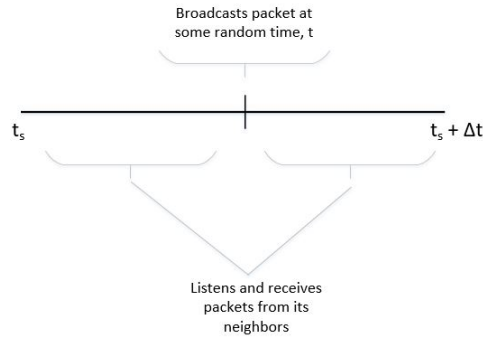


Figure 5.6: Ratio-Consensus Process per Iteration

At the given start time, all nodes in the network begin ratio-consensus simultaneously. At each iteration k , each node has two functions: (1) At

some random time t within Δt , a node i sends a packet to its neighbors containing $\mu_i[k+1]$ and $\sigma_i[k+1]$; and (2) the rest of the time within Δt , it listens and receives packets from its neighbors. This process is illustrated in Figure 5.6.

5.3.4 Clock Synchronization Protocol

The ratio-consensus algorithm relies on the update of two internal states, y_i and z_i , as discussed earlier. We assume that all participating nodes update their states in unison. However, in an actual implementation of this algorithm on a distributed communication network this assumption is not necessarily true, and considering this is a one-way form of communication (participating nodes do not send an acknowledgement packet once they receive a packet from its neighbors), there is no way to confirm each node's states are updated at the same iteration k .

Although the robust ratio consensus provides tolerance to packet drops, it does not account for this issue which could lead to convergence to an inaccurate solution or no convergence at all. To prevent this, all nodes need to be synchronized to a common time reference before the algorithm is executed. The synchronization protocol used in our hardware testbed is based on a hierarchy referencing time synchronization protocol [7]. This protocol is a three-step process that allows nodes to synchronize to a common clock reference. In the case of our hardware testbed, each node has a microcontroller with its own internal clock and we use the internal clock of the leader node as the common clock reference.

The leader node initiates the synchronization process by broadcasting a SYNC packet to all the follower nodes in the network at time t_1 . This SYNC packet contains the time the packet is sent, t_1 , as well as the address of a target node which it randomly chooses among its neighbors. The follower nodes not chosen as the target node (non-target node) receive this packet and take note of the time it was received, i.e. t_2^j , $j \rightarrow$ non-target node. The target node which receives the packet at some time t_2 , is defined by

$$t_2 = t_1 + d_1 + d_2, \quad (5.6)$$

where d_1 is the message propagation delay and d_2 is the local clock offset

between the target node and the leader node. The target node acknowledges that it received the packet by sending an ACK packet to the leader node at time t_3 . This ACK packet contains the times t_2 and t_3 . The leader node then receives this packet at some time t_4 defined as

$$t_4 = t_3 + d_1 - d_2. \quad (5.7)$$

The purpose of the first two steps of communication was to determine the unknown variables d_1 and d_2 . With the information the leader node now has, it can estimate these unknown variables as follows:

$$d_1 = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \quad (5.8)$$

$$d_2 = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}. \quad (5.9)$$

Note that we assume the message propagation delay d_1 is the same in both steps, but that is not the case in a communication network. The difference though is quite small so we choose to ignore it in our calculations. The leader node then broadcasts the final packet, which contains d_2 and t_2 , to all the follower nodes in the network. The follower nodes determine the offset of their clocks from that of the leader node, which acts as the reference clock. The target node offset is simply d_2 while the nontarget follower nodes determine their offset as follows:

$$\text{offset}_j = d_2 + t_2^j - t_2 \quad (5.10)$$

where the difference between the leader node, target node and nontarget nodes is taken into account. Figure 5.7 illustrates this process for a simple two-node case.

5.3.5 Modbus TCP Protocol

Between the hardware controllers and Typhoon HIL system we establish a client/server communication architecture connected on an Ethernet TCP/IP network. On this Ethernet TCP/IP network, we use the modbus TCP protocol due to its compatibility with the external communication features of Typhoon HIL and its popularity as an industrial serial communication protocol standard [13]. Modbus TCP operates using the client/server architecture

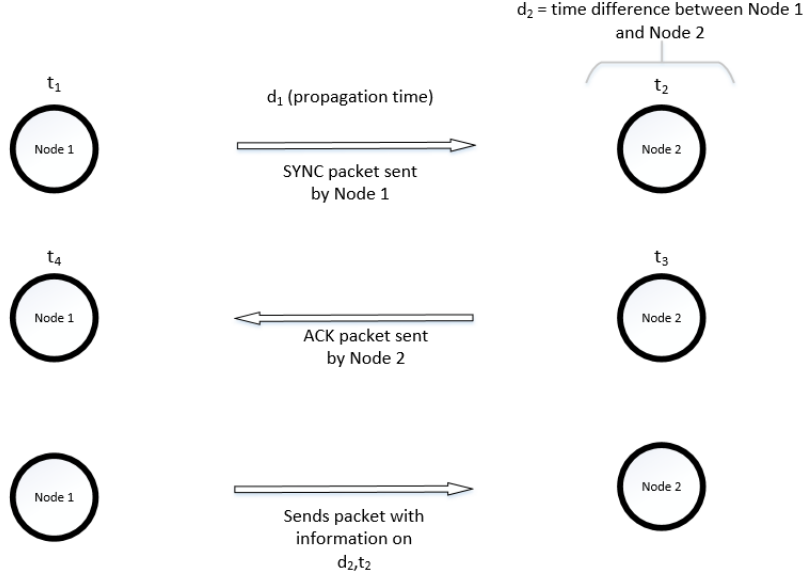


Figure 5.7: Synchronization Process for Two-Node Case

where clients can willingly write and read information to registers on a server but the server is unable to do that and simply waits on the client to request an action. The clients in this case are the hardware controllers while the server is the Typhoon HIL system as illustrated in Figure 5.8. This setup was ideal since the hardware controllers work independently from the Typhoon HIL and only communicate when they have completed the ratio-consensus process and the computed results are ready to be sent.

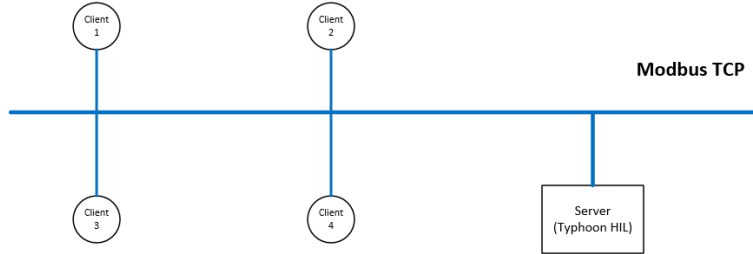


Figure 5.8: Communication Architecture

Each hardware controller is assigned an IP address, while within the Typhoon HIL system there is an assigned modbus device for each controller with its own unique IP address. Thus, if a controller sends a modbus TCP packet to the Typhoon HIL system, its IP address is known as the source address while the assigned modbus device IP address will be the destination address. The modbus TCP packet takes the form of either a request packet

(sent by client) or response packet (sent by server), and is encapsulated in the form of an application data unit (ADU) as shown in Figure 5.9. The MBAP header contains the transaction identifier, protocol identifier, length, and unit identifier. The transaction identifier is used for transaction pairing between client and server, the protocol identifier is used for intra-system multiplexing (set to 0 for modbus protocol), the length field is a byte count for the following fields, and the unit identifier is used for intra-system routing but is not necessary for our application.

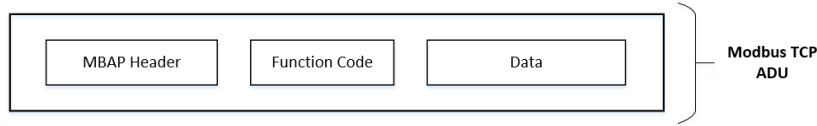


Figure 5.9: Application Data Unit

There are 4 types of registers defined in the modbus protocol as shown in the Table 5.1. With these registers, there are a set of functions that can be used to read/write data as defined in Table 5.2.

Table 5.1: Modbus Register Types

Register Type	Access	Size
Coil (Discrete Output)	read/write	1 bit
Discrete Input	read	1 bit
Input Registers	read	16 bits
Holding Registers	read/write	16 bits

Table 5.2: Function Codes

Function Code	Function Name
1	Read Coil
2	Read Discrete Input
3	Read Holding Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Holding Register
15	Write Multiple Coils
16	Write Multiple Holding Registers

In the Typhoon HIL system we configure each modbus device to have a certain number of registers with each register having a specific address. On

the hardware controller end, we define an array of 16-bit registers where stored information can be sent to the registers on the Typhoon HIL system using the function codes mentioned above with specified addresses. This allows us to send 32-bit integer data from the controllers to the Typhoon HIL system. Once the ratio-consensus process is complete, the final result is a floating point value that lies between 0 and 1. This is converted to a 32-bit integer by scaling it with a base value and then is sent from the controllers as a modbus request packet using two 16-bit registers with the 32-bit integer broken into two 16 bit values (upper 16 bits and lower 16 bits). Function code 16 is used to write this information to two 16-bit registers on the Typhoon HIL system which it then reconstructs to the original floating point value. After the assigned modbus device on the Typhoon HIL system has received the request packet, it then sends a response packet confirming it has received the packet. The general transaction process is illustrated in Figure 5.10.

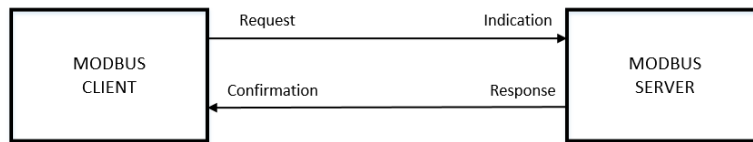


Figure 5.10: Client/Server Communication

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 Resource Allocation for 4-Node Case

In this experiment, we first show the effectiveness of our distributed algorithm implemented in our hardware testbed for a 4 node case by sending a single regulation signal used to adjust the active power setpoints of the DERs in the microgrid model. Then, we stream a set of regulation signals (4 seconds per regulation signal) over a 40 minute period.

We have a 4-node network set up as a ring communication graph, as illustrated in Figure 6.1. Each node has its own capacity limits, which determine resource distribution among nodes.

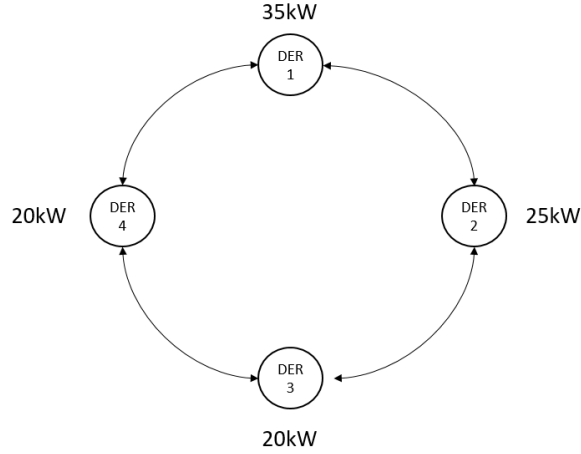


Figure 6.1: 4 Node Communication Graph

The initial values of y and z for each node are set to

$$y[0] = [0.5, 0, 0, 0],$$

$$z[0] = [0.35, 0.25, 0.2, 0.2],$$

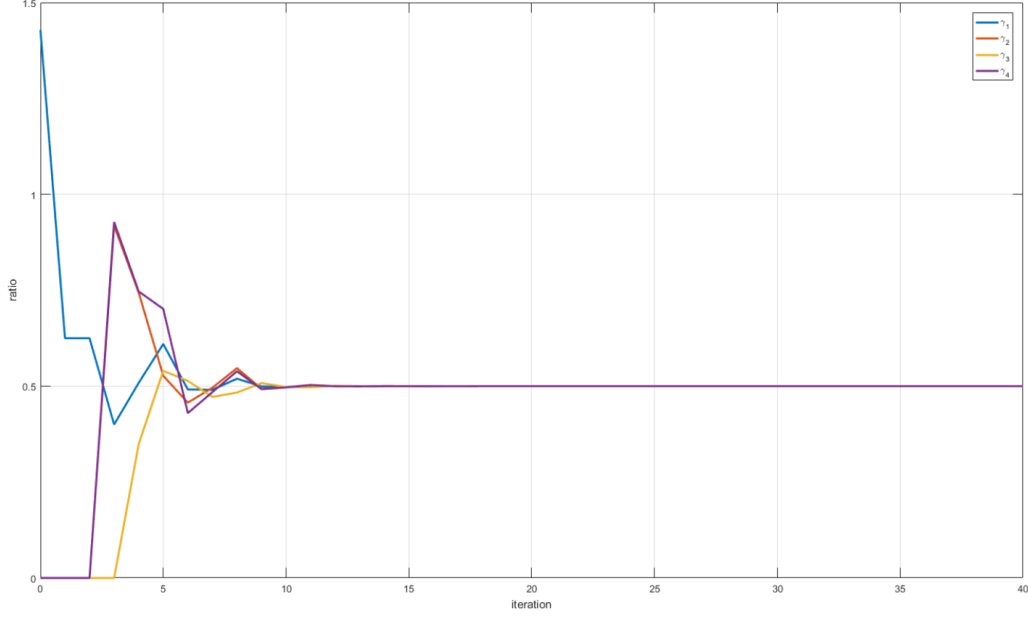


Figure 6.2: Evolution of γ for 4 nodes (Hardware Testbed)

where $y_i[0]$ and $z_i[0]$ correspond to the initial values of DER (node) i . The term $y_1[0]$ represents the required regulation power to be provided while $z_i[0]$ represents capacity limits of node i ; both values are normalized based on the total regulation capacity of the microgrid. The initial setpoints of the DERs in the microgrid are set to zero so the maximum and minimum total regulation capacity is defined as

$$P_{max} = \sum_{i=1}^4 z_i[0] = 1,$$

$$P_{min} = 0.$$

Each γ_i should converge to the same solution, $\gamma = \frac{\sum_{j=1}^4 y_j[0]}{\sum_{j=1}^4 z_j[0]} = 0.5$; this is shown to be the case in Figure 6.2 where the plotted results of γ_i from our hardware testbed are illustrated.

Using these results, the resource contribution by each DER is determined as follows:

$$\Delta P_i = \gamma_i * z_i[0]. \quad (6.1)$$

When compared to the MATLAB simulation results as shown in Figure

6.3, it is apparent that it takes longer to converge in our hardware testbed. This is due to the effect of packet drops which slow down the algorithm, but because we use the robust ratio-consensus algorithm we are still able to achieve convergence.

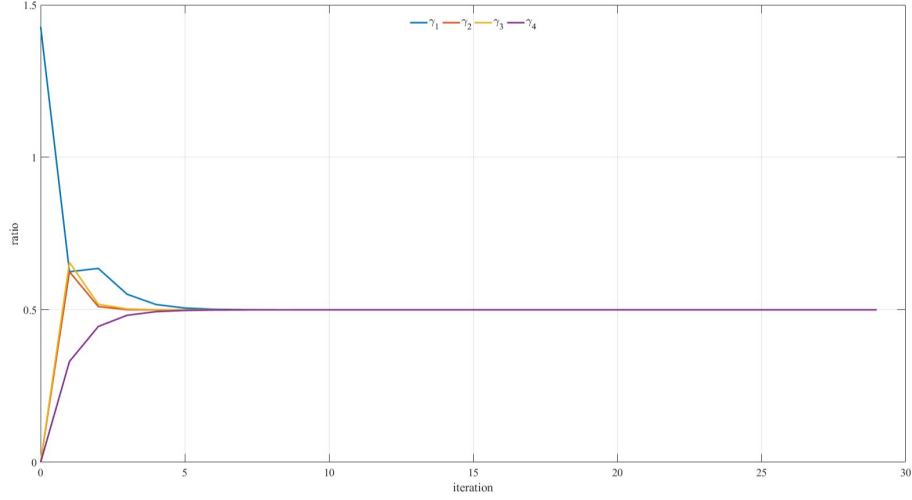


Figure 6.3: Evolution of γ for 4 Nodes (Simulation)

In the second case each node has the same capacity (35 kW) and we continuously stream regulation signals (updated every 4 seconds) to the leader node over a 40 minute period. Figure 6.4 illustrates the frequency regulation provision process.

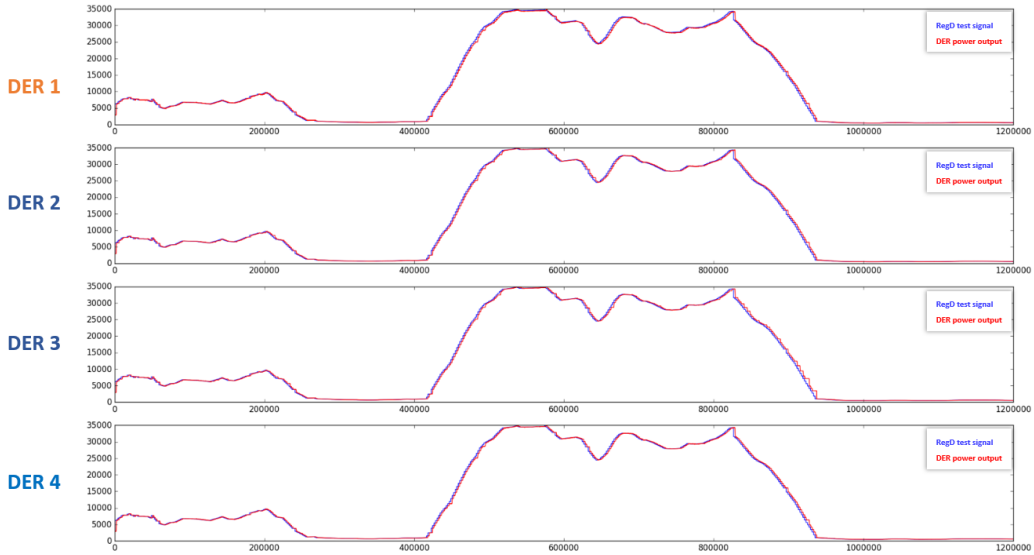


Figure 6.4: RegD Signal vs. DER Power Output

6.2 Resource Allocation for 6-Node Case

In this experiment, we first show the effectiveness of our distributed algorithm implemented in our hardware testbed for a 6-node case and then show the frequency regulation provision process for a certain period of time. In this case, however, our physical layer is not the same microgrid model as discussed earlier due to the processing constraints of the Typhoon HIL 402 system and specific requirements. With regard to the specific requirements, each cyber node needs to control 5 HIL simulated power nodes which goes beyond what the Typhoon HIL 402 can handle. Instead each power node is represented by a signal controlled voltage source connected to a fixed impedance. When the regulation power is distributed accordingly in the cyber layer, each cyber node distributes the power to be provided evenly by changing the voltage.

We have a 6-node network set up as a ring communication graph, as illustrated in Figure 6.5. Each node has its own capacity limits which determine resource distribution among nodes.

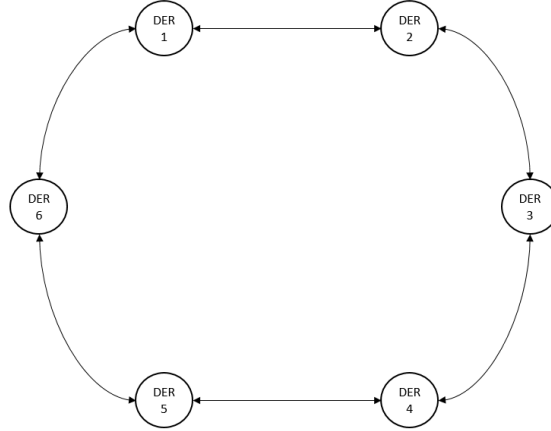


Figure 6.5: 6-Node Communication Graph

The initial values of y and z for each node are set to

$$y[0] = [0.72, 0, 0, 0, 0, 0],$$

$$z[0] = [0.5, 0.35, 0.25, 0.2, 0.1, 0.1],$$

where $y_i[0]$ and $z_i[0]$ correspond to the initial values of node i . The term $y_1[0]$ represents the required regulation power to be provided, while $z_i[0]$

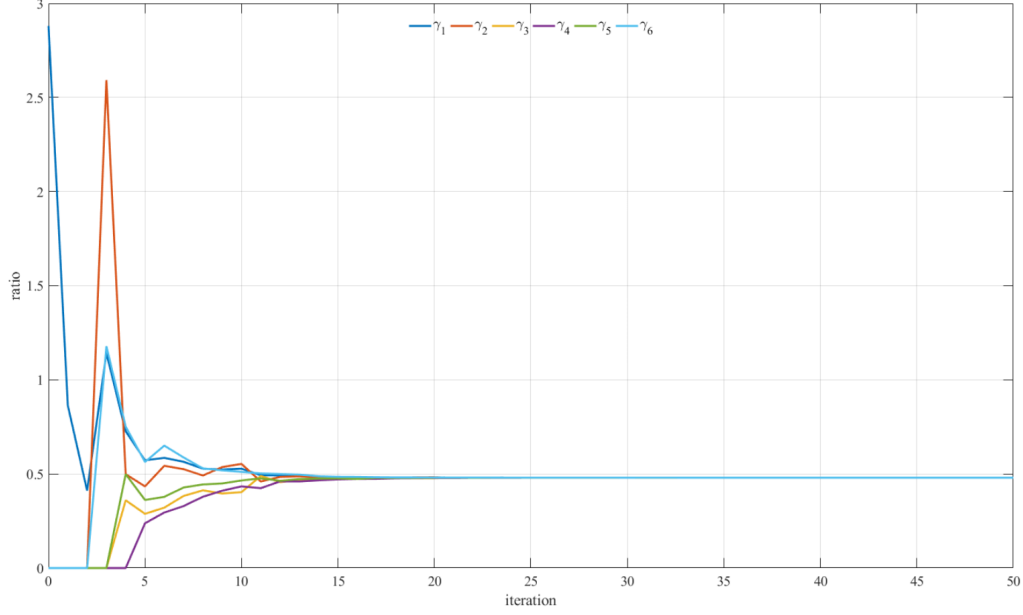


Figure 6.6: Evolution of γ for 6 nodes (Hardware Testbed)

represents capacity limits of node i ; both values are normalized based on the total regulation capacity of the microgrid. The initial setpoints of the DERs in the microgrid are set to zero so the maximum and minimum total regulation capacity is defined as

$$P_{max} = \sum_{i=1}^6 z_i[0] = 1.5,$$

$$P_{min} = 0.$$

Each γ_i should converge to the same solution, $\gamma = \frac{\sum_{j=1}^4 y_j[0]}{\sum_{j=1}^4 z_j[0]} = 0.48$; this is shown to be the case in Figure 6.6 where the plotted results of γ_i from our hardware testbed are illustrated.

Using these results, the resource contribution by each node is determined as follows:

$$\Delta P_i = \gamma_i * z_i[0]. \quad (6.2)$$

In the second case, we continuously stream regulation signals to the leader node which initiates the ratio-consensus process in the network. This experiment lasts for around 1400 seconds where at each round of ratio consensus,

a new regulation signal is received by the leader node. Figures 6.7 - 6.12 illustrate the reference tracking by each cyber node. Upon observation, one will see that each node has the same distribution but at different scales due to the capacities of the different cyber nodes.

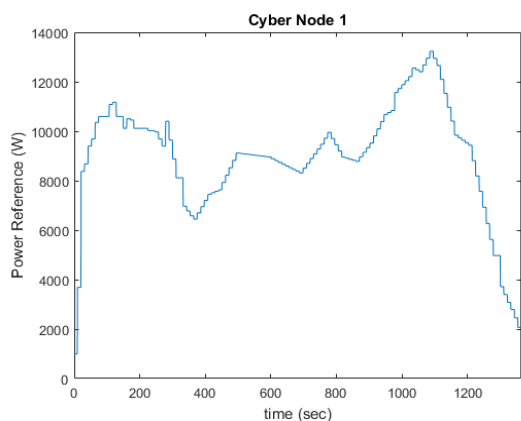


Figure 6.7: Node 1 Reference Tracking

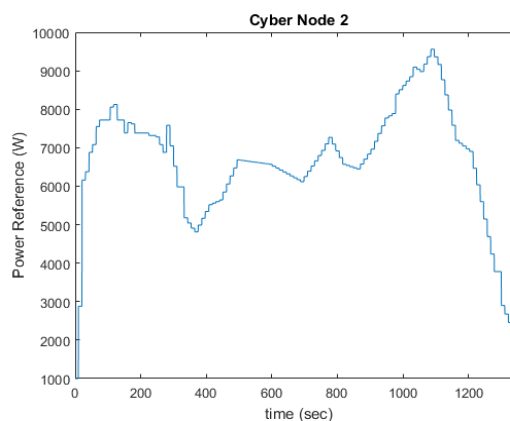


Figure 6.8: Node 2 Reference Tracking

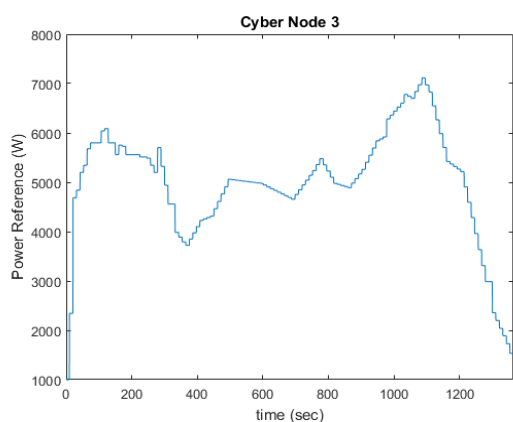


Figure 6.9: Node 3 Reference Tracking

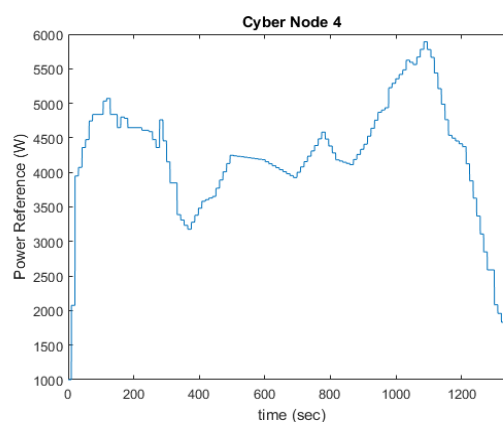


Figure 6.10: Node 4 Reference Tracking

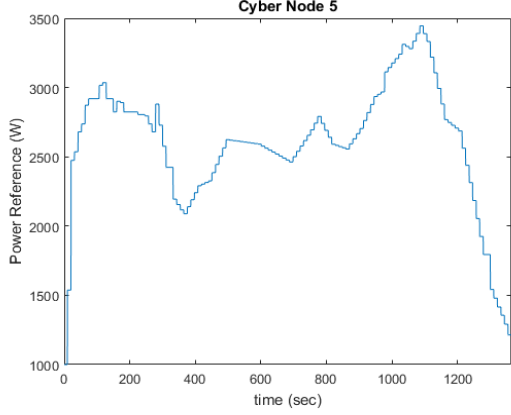


Figure 6.11: Node 5 Reference Tracking

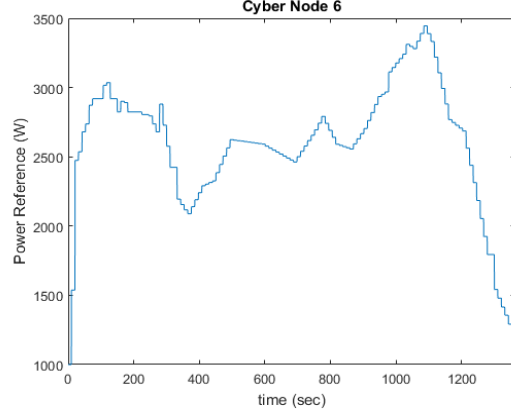


Figure 6.12: Node 6 Reference Tracking

6.3 Resource Allocation to Minimize System Losses

In our previous experiments, we allocate resources based on capacity limits but neglect system losses that occur in our microgrid. Thus we are not providing the full regulation power requested. In this experiment, our objective is to minimize incremental total network losses [14], which are approximated by a linear function of the incremental changes in active power provided by the DERs. This is an optimization problem that is presented as follows:

$$\begin{aligned}
 & \text{minimize} && \sum_{j=1}^N \Lambda_j x_j \\
 & \text{subject to} && \sum_{j=1}^N (1 - \Lambda_j) x_j = X, \\
 & && \underline{x}_j \leq x_j \leq \bar{x}_j, \forall j \in \mathcal{X}.
 \end{aligned} \tag{6.3}$$

In (6.3), $\mathcal{X} = \{1, \dots, N\}$ denotes the set of DERs in the microgrid; X denotes the regulation signal received from the aggregator; x_j , denotes the regulation power provided by DER j ; \underline{x}_j and \bar{x}_j denote the minimum and maximum values of x_j respectively; and $\Lambda_j < 1$ is the loss factor associated with DER j . This loss factor is defined to be the partial derivative of the total network losses with respect to active power injection from DER j evaluated at the operating point corresponding to the DER's nominal active power outputs. This value is obtained via a measurement-based approach (see, e.g. [15]), but in this experiment, we assume the nodes already know their loss factors for a given operating point.

A centralized solution is presented first, followed by the distributed version

which is implemented in our hardware testbed.

The lagrange dual problem of (6.3) is

$$\max_{\mu} \inf_{x_j \in [\underline{x}_j, \bar{x}_j]} \sum_{j=1}^N \Lambda_j x_j - \mu \left(\sum_{j=1}^N (1 - \Lambda_j) x_j - X \right), \quad (6.4)$$

which can be rewritten as

$$\max_{\mu} \mu X - \sum_{j=1}^N f_j(\mu), \quad (6.5)$$

where $f_j(\mu)$, $\forall j$, is a piecewise linear function of the following form:

$$f_j(\mu) = \begin{cases} (1 - \Lambda_j) \underline{x}_j \mu - \Lambda_j \underline{x}_j, & \mu \leq \mu_j, \\ (1 - \Lambda_j) \bar{x}_j \mu - \Lambda_j \bar{x}_j, & \mu > \mu_j, \end{cases} \quad (6.6)$$

where $\mu_j = \frac{\Lambda_j}{1 - \Lambda_j}$, $j \in \mathcal{X}$, and defines a multiset $\mathcal{M} = \{\mu_1 \cdots, \mu_N\}$.

If the primal problem is feasible then, by strong duality, there exists a solution μ^* that solves (6.5) which we show below. Let $h(\mu) := \sum_{j=1}^N h_j(\mu)$, where $h_j(\mu)$ is a two-valued step function defined as follows:

$$h_j(\mu) = \begin{cases} (1 - \Lambda_j) \underline{x}_j, & \text{if } \mu \leq \mu_j, \\ (1 - \Lambda_j) \bar{x}_j, & \text{if } \mu > \mu_j. \end{cases} \quad (6.7)$$

Under the assumption that (6.3) is feasible, we can compute μ^* as follows:

$$\mu^* = \arg \min_{\mu \in \mathcal{M}, \frac{h(\mu)}{X} \leq 1} \left| \frac{h(\mu)}{X} - 1 \right|. \quad (6.8)$$

Given μ^* , we can then determine the optimal solution to the primal problem. First, partition \mathcal{X} as follows:

$$\begin{aligned} \mathcal{X}^+ &= \{j \in \mathcal{X} : \mu_j > \mu^*\}, \\ \mathcal{X}^0 &= \{j \in \mathcal{X} : \mu_j = \mu^*\}, \\ \mathcal{X}^- &= \{j \in \mathcal{X} : \mu_j < \mu^*\}. \end{aligned} \quad (6.9)$$

The DERs in \mathcal{X}^+ (\mathcal{X}^-) are the more (less) “costly” units (also referred to as

the non-marginal DERs), and their outputs are set as follows:

$$x_j^* = \begin{cases} \underline{x}_j, & \forall j \in \mathcal{X}^+, \\ \bar{x}_j, & \forall j \in \mathcal{X}^-. \end{cases} \quad (6.10)$$

The DERs in \mathcal{X}^0 are referred to as the marginal DERs. If \mathcal{X}^0 is a singleton, it follows from the equality constraint in (6.3) that the output of DER $j \in \mathcal{X}^0$ must be set to:

$$x_j = \frac{X - \sum_{l \in \mathcal{X}^+} (1 - \Lambda_l) \underline{x}_l - \sum_{l \in \mathcal{X}^-} (1 - \Lambda_l) \bar{x}_l}{1 - \Lambda_j}. \quad (6.11)$$

If there is more than one marginal DER, then the solution may not be unique. Thus we introduce a fair splitting policy where resource allocation among marginal DERs takes place as follows:

$$x_j^* = \underline{x}_j + \alpha(\bar{x}_j - \underline{x}_j), \quad (6.12)$$

where

$$\alpha = \frac{X - \sum_{l \in \mathcal{X}^+ \cup \mathcal{X}^0} (1 - \Lambda_l) \underline{x}_l - \sum_{l \in \mathcal{X}^-} (1 - \Lambda_l) \bar{x}_l}{\sum_{l \in \mathcal{X}^0} (1 - \Lambda_l)(\bar{x}_l - \underline{x}_l)}. \quad (6.13)$$

Thus solving this optimization problem requires two main steps: first finding μ^* using equation (6.8), then computing x_j , $\forall j \in \mathcal{X}$ using (6.10) and (6.12).

When solving this problem using our distributed communication network, we assume that Λ_1 to Λ_N , and correspondingly \mathcal{M} , are known to all nodes. Then to determine which $\mu \in \mathcal{M}$ is the optimal choice, we use (6.8). To do this on our distributed communication network, each node needs to know $\frac{h(\mu)}{X}$ for $\forall \mu \in \mathcal{M}$ to determine which μ minimizes $|\frac{h(\mu)}{X} - 1|$. This is achieved using a slightly modified version of the ratio consensus algorithm where

$$\begin{aligned} y_j^{(i)}[0] &= h_j(\mu_i), \forall i \in \mathcal{X}, \\ z_1[0] &= X, \\ z_j[0] &= 0, \forall j \in \mathcal{X} \setminus \{1\}. \end{aligned} \quad (6.14)$$

Thus instead of just having two internal states, each node has $j + 1$ internal states. So after a certain number of iterations, each node will asymptotically

learn $\frac{h(\mu_i)}{X}$, $\forall i \in \mathcal{X}$, as follows:

$$\gamma_j^{(i)} = \frac{h(\mu_i)}{X} = \frac{\sum_{l \in \mathcal{X}} h_l(\mu_i)}{X} = \lim_{k \rightarrow \infty} \frac{y_j^{(i)}[k]}{z_j[k]}. \quad (6.15)$$

Once the optimal μ has been determined, we now know the marginal and non-marginal DERs. We then proceed to calculate α as shown in equation (6.13) for the marginal DERs by using the normal ratio consensus process as follows:

$$\begin{aligned} y_1[0] &= X - (1 - \Lambda_1)\underline{x}_1, \text{ if } 1 \in \mathcal{X}^+ \cup \mathcal{X}^0, \\ y_1[0] &= X - (1 - \Lambda_1)\bar{x}_1, \text{ if } 1 \in \mathcal{X}^-, \\ y_j[0] &= -(1 - \Lambda_j)\underline{x}_j, \forall j \in \mathcal{X}^+ \cup \mathcal{X}^0, \\ y_j[0] &= -(1 - \Lambda_j)\bar{x}_j, \forall j \in \mathcal{X}^-, \\ z_j[0] &= (1 - \Lambda_j)(\bar{x}_j - \underline{x}_j), \forall j \in \mathcal{X}^0, \\ z_j[0] &= 0, \forall j \notin \mathcal{X}^0. \end{aligned} \quad (6.16)$$

We now show an experiment for a 4-node ring communication graph where resource allocation to minimize system losses is achieved.

The DER maximum regulation power and minimum regulation power are as follows: $\bar{x}_1 = 0.3$, $\bar{x}_2 = 0.8$, $\bar{x}_3 = 0.5$, $\bar{x}_4 = 0.4$, and $\underline{x}_1 = -0.3$, $\underline{x}_2 = -0.8$, $\underline{x}_3 = -0.5$, $\underline{x}_4 = -0.4$. The LFs are as follows: $\Lambda_1 = 0$, $\Lambda_2 = 0.01$, $\Lambda_3 = 0.02$, $\Lambda_4 = 0.04$; as a result, $\mathcal{M} = \{0, \frac{1}{99}, \frac{1}{49}, \frac{1}{24}\}$. And the leader node receives a regulation signal, $X = 1.8$, from the aggregator. The consensus results when node 1 computes μ^* are shown in Figure 6.13. The ratios $\gamma_j^{(i)}$ are approximate values for $\frac{h(\mu_i)}{X}$. The estimated values of $\frac{h(\mu_1)}{X}$, $\frac{h(\mu_2)}{X}$, $\frac{h(\mu_3)}{X}$, $\frac{h(\mu_4)}{X}$ by all nodes converge to -1.0922 , -0.7590 , 0.1210 , 0.6660 , respectively, after 35 iterations.

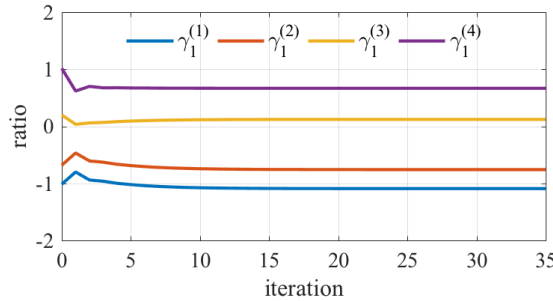


Figure 6.13: Evolution of Ratio-Consensus when Node 1 Computes μ^*

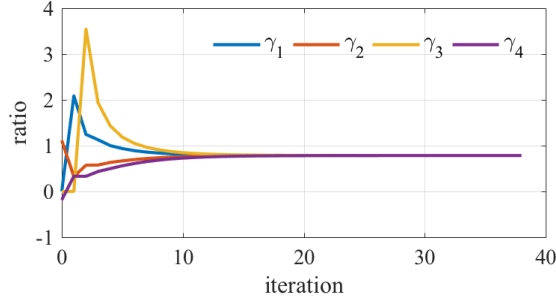


Figure 6.14: Evolution of Ratio-Consensus when Computing α

Based on this result, all nodes will agree on the value of μ^* to be $\frac{1}{24}$. This result is indeed intuitively correct since DER 4 has the largest LF, yet the maximum total regulation power provided by DERs 1 - 3 cannot meet the regulation requirement. Thus after each node learns μ^* , the regulation power of DERs 1 to 3 will be set to their corresponding maxima while DER 4, which is the marginal DER, will set its regulation power based on the second round of ratio-consensus which essentially computes α as discussed earlier. The consensus results when computing α is shown in Figure 6.14.

All nodes reach an agreement on the value of α to be 0.7840 after 38 iterations. As such, the regulation power of DER 4 will be set to 0.2272, which is computed based on (6.12). It can be easily verified that the total provided regulation power in the presence of losses is 1.8 when DERs set their regulation power to 0.3, 0.8, 0.5, and 0.2272, respectively.

CHAPTER 7

CONCLUSION

In this thesis, we presented a hardware-in-the-loop platform used to test the effectiveness of distributed algorithms for coordinating the response of DERs so as to provide a certain service to the bulk grid. We first discussed one such service, frequency regulation, highlighting the benefits it provides and how an entity participates in the frequency regulation provision process. We then presented an overview of our hardware testbed, showing how the different components are integrated together and giving a general idea of the experimental process. This hardware testbed is broken down into two aspects: physical layer and cyber layer. The physical layer is represented by a microgrid modeled in the Typhoon HIL system. A description of the DERs in the microgrid is provided along with the communication interface that connects the DER to a dedicated hardware controller which exists in the cyber layer. Within the cyber layer, there is a computing platform acting as the aggregator which communicates via USB with a hardware controller known as the leader node. This leader node is just one of many other controllers (known as follower nodes) which are all linked wirelessly and form a network that enables the application of a distributed control scheme. The components that make up each hardware controller are described appropriately with their individual functionality highlighted. The software used in these hardware controllers is discussed, with special emphasis placed on the distributed control algorithm and its implementation in software.

Using this hardware testbed, we show the effectiveness of a distributed algorithm referred to as ratio-consensus in coordinating DERs to provide frequency regulation services. In our experiments, we discuss several results showing resource allocation among the DERs through practical convergence of ratio-consensus along with power reference tracking of the DERs in order to meet the regulation objective. We also present resource allocation among DERs where the goal is to minimize system losses in order to better satisfy

the regulation objective. In conclusion, we are able to use our hardware-in-the-loop platform to provide frequency regulation services and show the benefits our application can provide to the bulk grid.

REFERENCES

- [1] E. Hossain, E. Kabalci, R. Bayindir, and R. Perez, “Microgrid testbeds around the world: State of art,” *Energy Conversion and Management*, vol. 86, no. Supplement C, pp. 132 – 153, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0196890414004233>
- [2] S. T. Cady, A. D. Domínguez-García, and C. N. Hadjicostis, “A distributed generation control architecture for islanded ac microgrids,” *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1717–1735, Sept 2015.
- [3] Typhoon HIL Inc. website. [Online]. Available: <https://www.typhoon-hil.com/>
- [4] J. Meng, Y. Wang, C. Wang, and H. Wang, “Design and implementation of hardware-in-the-loop simulation system for testing control and operation of dc microgrid with multiple distributed generation units,” *IET Generation, Transmission Distribution*, vol. 11, no. 12, pp. 3065–3072, 2017.
- [5] Energy Storage Association, “Grid operation benefits.” [Online]. Available: <http://energystorage.org/energy-storage/energy-storage-benefits/benefit-categories/grid-operations-benefits>
- [6] A. D. Domínguez-García and C. N. Hadjicostis, “Distributed algorithms for control of demand response and distributed energy resources,” in *Proc. of IEEE Conference on Decision and Control and European Control Conference*, Dec 2011, pp. 27–32.
- [7] H. Dai and R. Han, “TSync: a lightweight bidirectional time synchronization service for wireless sensor networks,” *Mobile Computing and Communications Review*, vol. 8, pp. 125–139, 2004.
- [8] “Typhoon HIL 402 hardware-in-the-loop system.” [Online]. Available: <https://www.typhoon-hil.com/doc/products/Typhoon-HIL402-brochure.pdf>
- [9] R. W. Erickson and D. Maksimovic, *Fundamentals of Power Electronics*. Springer, 2001.

- [10] M. Prodanovic and T. C. Green, “Control and filter design of three-phase inverters for high power quality grid connection,” *IEEE Transactions on Power Electronics*, vol. 18, no. 1, pp. 373–380, Jan 2003.
- [11] Processing Software website. [Online]. Available: <https://processing.org/>
- [12] S. T. Cady, A. Domínguez-García, and C. Hadjicostis, “Robust implementation of distributed algorithms for control of distributed energy resources,” in *North American Power Symposium*, 09 2011, pp. 1 – 5.
- [13] RTA Inc., “Modbus TCP/IP overview.” [Online]. Available: <https://www.rtaautomation.com/technologies/modbus-tcpip/>
- [14] H. Xu, S. C. Utomi, A. D. Domínguez-García, and P. W. Sauer, “Coordination of distributed energy resources in lossy networks for providing frequency regulation,” in *Proc. of IREP Bulk Power System Dynamics and Control Symposium*, August 2017.
- [15] Y. C. Chen, A. D. Domínguez-García, and P. W. Sauer, “Measurement-based estimation of linear sensitivity distribution factors and applications,” *IEEE Transactions on Power Systems*, vol. 29, no. 3, pp. 1372–1382, May 2014.