A Distributed Generation Control Architecture for Small-Footprint Power Systems

Stanton T. Cady, *Student Member, IEEE*
Alejandro D. Domínguez-García, *Member, IEEE*
Christoforos N. Hadjicostis, *Senior Member, IEEE*

July 2013

Abstract

In this paper, we propose a distributed architecture for generation control in small-footprint power systems. Although they are smaller and have lower ratings, the generation control objectives for a small-footprint power system are similar to those in larger counterparts, e.g., bulk-power transmission networks. However, in large power systems, the implementation of the generation control functions is centralized, i.e., there is a computer that resides in a centralized location, e.g., a control center, with measurements and control signals telemetered to and from the generating units and the centrally-located computer. The architecture for generation control that we propose in this paper does not rely on such a centrally-located computer. Instead, the implementation of the control functions is distributed and relies on low-complexity iterative algorithms that only use local measurements and certain information acquired through exchange of information with neighboring generating units. We provide analytical and experimental results that verify the effectiveness of the proposed architecture for generation control in small-footprint power systems, and illustrate the performance of the aforementioned distributed algorithms under a variety of scenarios.

S. T. Cady and A. D. Domínguez-García are with the ECE Department at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: {scady2, aledan}@ILLINOIS.EDU.

C. N. Hadjicostis is with the ECE Department at the University of Cyprus, Nicosia, Cyprus, and also with the ECE Department at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: chadjic@UCY.AC.CY.

The work of S. T. Cady and A. D. Domínguez-García was supported in part by the National Science Foundation (NSF) under grant ECCS-CPS-1135598. The work of C. N. Hadjicostis was supported in part by the European Commission (EC) Seventh Framework Programme (FP7/2007-2013), under grant agreements INFSO-ICT-223844 and PIRG02-GA-2007-224877. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of NSF or EC.

I. INTRODUCTION

While not strictly defined, we consider any group of interconnected loads and synchronous generators to be a small-footprint power system as long as, relative to large power systems such as bulk-power transmission networks, they are physically smaller and have lower ratings and capacity. Although the archetypical example is a land-based collection of small generation units and loads which must be capable of operating in isolation from the utility grid, i.e., a microgrid (see, e.g., [1]–[4]), small-footprint power systems can also be found in applications such as more-electric aircrafts (see, e.g., [5], [6]), ships with electric propulsion (see, e.g., [7], [8]), and electrical installations for supporting telecommunication equipment (see, e.g., [9]).

Despite the aforementioned differences in size, the generation control objectives for small-footprint power systems are similar to their larger counterparts. In particular, for any power system: (i) generation and demand must be constantly balanced, (ii) the frequency must be regulated, and (iii) the costs of generation must be minimized. In a large power system, these objectives are achieved through a three-layered generation control architecture. The bottom layer operates nearly instantaneously and is responsible for balancing generation and demand. The primary function of the middle layer is to regulate frequency, i.e., ensure the system operates as close as possible to the nominal frequency value, e.g., 60 Hz. The function of the top layer is to optimally dispatch the generating units, i.e., ensure that the units are producing power in such a way that the total cost of operation is minimum [10].

In a typical large power system, among the three previously-mentioned control layers, only the bottom one operates using completely local information; for the top two layers, a centralized decision maker is required to coordinate and control the generators. While the characteristics of large power systems necessitate this top-down approach to generation control, small-footprint power systems are amenable to a control architecture in which the decision-making is distributed among controllers located at the generating units. Through the exchange of information among local controllers, this alternative architecture can replicate the functionality of traditional generation control systems without requiring full knowledge of the number, type, or capabilities of generating units in the system; or a communication network connecting each generator to a central processor. Furthermore, compared to centralized ones, a distributed generation control approach can more easily adapt to changes, allowing the system to operate regardless of additions or removals of generating units.

The main contribution of this work is the development and implementation of a control architecture for small-footprint power systems, which replicates the functionality of the aforementioned top two control layers, i.e., frequency regulation and optimal dispatch, in a distributed fashion. In our setup, we assume that each generator is equipped with a local controller and a wireless transceiver through which information can be exchanged with neighboring generators. By relying on this communication network and on simple computations executed by the local controllers, we provide two algorithms that allow the generators to determine their output commands in order to regulate the system frequency and minimize total operational costs. Beyond introducing distributed alternatives to the top two control layers, a major difference between our distributed control architecture and centralized counterparts used in bulk power systems is that ours is event triggered, i.e., generator commands can be updated at non-fixed time instants, which allows a faster return to operation at nominal frequency.

In order to demonstrate the effectiveness of our distributed generation control architecture, we built an experimental microgrid. The electrical network of this microgrid is comprised of several interconnected small synchronous generators and resistive loads, whereas the cyber network for communication and control is comprised of Arduino microcontrollers outfitted with wireless transceivers. Using this microgrid, we experimentally verified the performance of the proposed control architecture under a variety of scenarios. Specifically, we provide results that illustrate the operation of the distributed frequency regulation and optimal dispatch functions following both an increase and a decrease in load. Additionally, we show how a generating unit acting as spinning reserve can independently determine if the collective capacity of the online units is not sufficient to balance the load, and act accordingly.

The remainder of this paper is organized as follows. In Section II, we introduce models to describe (i) the small-footprint power system dynamics, and (ii) the communication network describing the exchange of information among generating units in the system; additionally, we discuss a linear-iterative distributed algorithm that will be used as a primitive for implementing two algorithms on which the proposed control architecture relies. In Section III, we provide an overview of the desired control functions and outline specific control objectives. In Section IV, we describe the two algorithms that distributively implement the desired control functions. Section V describes a laboratory-grade microgrid that we developed for testing the performance of the proposed architecture. Experimental results obtained using the aforementioned microgrid are presented in Section VI. Concluding remarks are presented in Section VII.

II. Preliminaries

We begin this section by introducing a nonlinear differential algebraic equation (DAE) model to describe the dynamics in the electrical network of a small-footprint power system. Next, we introduce a graph-theoretic model to describe the cyber network for communication and computation which is responsible for controlling the electrical network. Figure 1 contains a block diagram that describes the interactions between these two networks; the components illustrated and the notation used in this diagram are introduced throughout this section and the next. Finally, we formulate a linear iterative algorithm that adheres to the

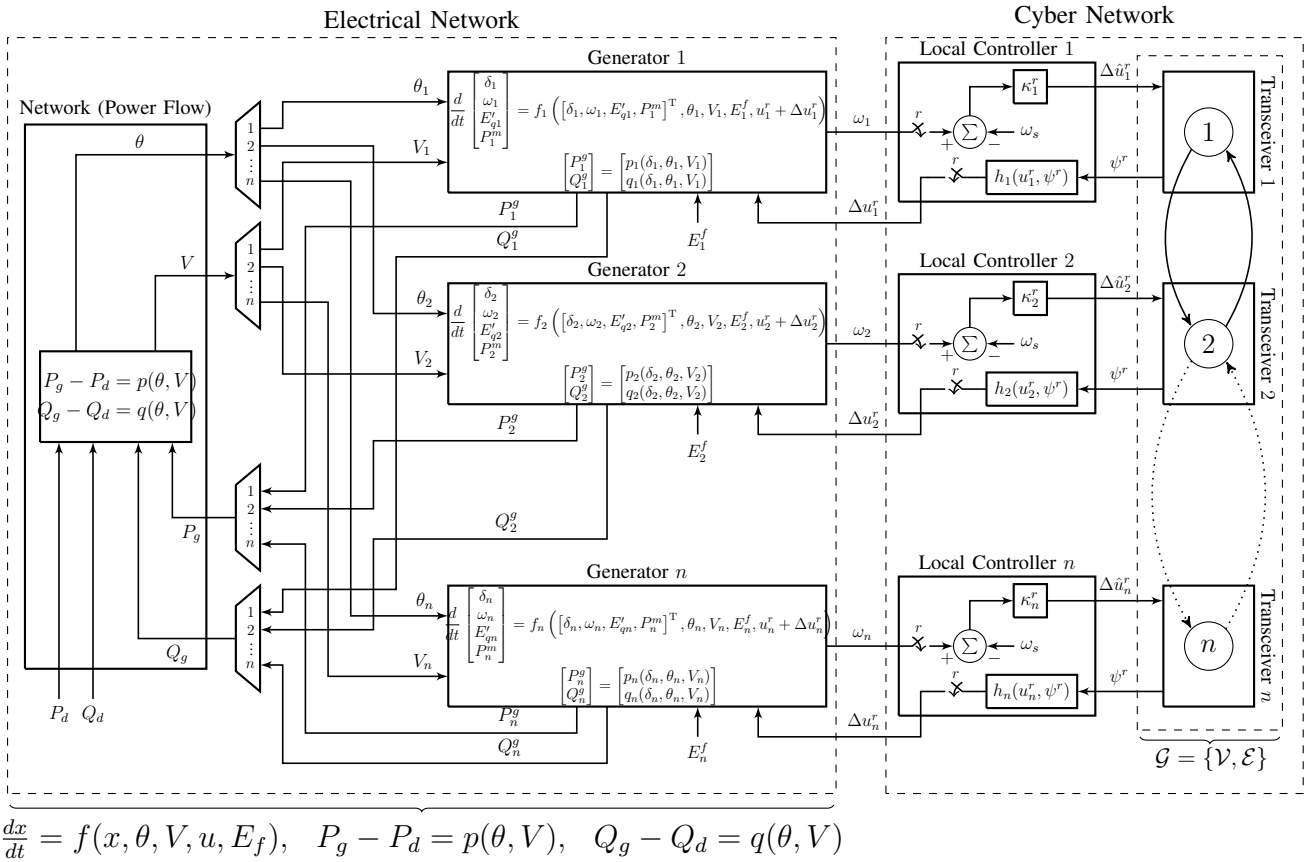


Fig. 1: Distributed generation control architecture block diagram: on the left, the connectivity between DGRs and loads is not explicitly shown; on the right, the communication topology linking the different transceivers is not explicitly shown. [The connections between the transceivers are not necessarily bidirectional, and the connectivity graph does not necessarily have a regular structure as perhaps implied by the figure; the only assumption is that the communication topology forms a strongly connected directed graph.]

aforementioned communication network, which will serve as a primitive for distributively implementing the frequency regulation and optimal dispatch functions.

A. Electrical Network Model

Similar to their larger counterparts, the electromechanical behavior of a small-footprint power system can be described by a DAE model, where the differential part describes the dynamics of synchronous generators and the algebraic part describes the power balance in the electrical network. For the systems under consideration in this work, we assume that each bus may have a synchronous generator, referred to as a distributed generation resource (DGR), a load, or both. Furthermore, we assume that the dynamics of each synchronous generator can be represented by the so-called *flux-decay model* with negligible stator resistance (see, e.g., [11]).

Assume that the system has n buses, indexed as $1, 2, \ldots, n$; then for bus i and its connected DGR, denote the rotor electrical angular position (with respect to a synchronous reference rotating at ω_s) as δ_i; the rotor electrical angular speed as ω_i; the quadrature-axis transient internal voltage as E'_{qi}; the primer mover mechanical power as P_i^m; the field voltage as E_i^f; the generation set-point as u_i; the magnitude of the bus voltage as V_i; and the angle of the bus voltage as θ_i. Let $x_i = \left[\delta_i, \omega_i, E'_{qi}, P_i^m\right]^{\mathrm{T}}$, $i = 1, \ldots, n$, and define $x = \left[x_1^{\mathrm{T}}, x_2^{\mathrm{T}}, \ldots, x_n^{\mathrm{T}}\right]^{\mathrm{T}}$. Additionally, define $\theta = \left[\theta_1, \theta_2, \ldots, \theta_n\right]^{\mathrm{T}}$, $V = \left[V_1, V_2, \ldots, V_n\right]^{\mathrm{T}}$, $u = \left[u_1, u_2, \ldots, u_n\right]^{\mathrm{T}}$, $E_f = \left[E_1^f, E_2^f, \ldots, E_n^f\right]$, $P_d = \left[P_1^d, P_2^d, \ldots, P_n^d\right]^{\mathrm{T}}$, and $Q_d = \left[Q_1^d, Q_2^d, \ldots, Q_n^d\right]^{\mathrm{T}}$, where P_i^d and Q_i^d are the real and reactive power demands at bus i, respectively. The overall system electromechanical dynamics can be described by

$$\frac{dx}{dt} = f(x, \theta, V, E_f, u), \tag{1}$$

$$P_g - P_d = p(\theta, V), \tag{2}$$

$$Q_g - Q_d = q(\theta, V), \tag{3}$$

with $f = [f_1, f_2, \ldots, f_n]^{\mathrm{T}}$, where $f_i : \mathbb{R}^4 \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^4$ is defined by the dynamics of the i^{th} DGR; and $p : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^n$ and $q : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^n$ are defined by the network power flow equations (see the left-hand side of Fig. 1 for a graphical depiction of the electrical network model). The reader is referred to [11] for a detailed discussion of the model in (1)–(3).

B. Cyber Network Model

In order to realize a distributed implementation of the frequency regulation and optimal dispatch functions, we assume that each DGR is outfitted with a local controller that is capable of performing simple computations. Additionally, we assume that each local controller is equipped with a transceiver through which information can be transmitted (possibly unidirectionally) to the local controllers of other nearby DGRs.

In this work, we describe the communication network interconnecting the DGR local controllers by a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} := \{1, 2, \ldots, n\}$ is the vertex set, with each vertex—or node—corresponding to a DGR; and where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, with the ordered pair $(i, j) \in \mathcal{E}$ if node i can receive information from node j; see the right-hand side of Fig. 1 for a graphical depiction of the cyber network model.By modeling the network using a directed graph, we allow for a very general communication modality with possibly asymmetric communication links; thus, we may have a flow of information from node i to node j and not the converse, i.e., $(i, j) \in \mathcal{E}$ while $(j, i) \notin \mathcal{E}$. For notational convenience, we require self-loops for all nodes, that is, $(i, i) \in \mathcal{E}$, $\forall i \in \mathcal{V}$.

We define the set of vertices from which the local controller of DGR i can receive information to be $\mathcal{N}_i^- := \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$. Similarly, we define the set of vertices to which i can send information to be $\mathcal{N}_i^+ := \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$, and we refer to $\mathcal{N}_i^-$ and $\mathcal{N}_i^+$ as the in- and out-neighborhood of node i, respectively. Furthermore, we denote the cardinality of the out-neighborhood, referred to as the

out-degree, of node i as $\mathcal{D}_i^+ := |\mathcal{N}_i^+|$. Note that as a consequence of the self-loop requirement, each vertex is a member of its own in- and out-neighborhood, i.e., $i \in \mathcal{N}_i^-$ and $i \in \mathcal{N}_i^+$, $\forall i \in \mathcal{V}$. Throughout the remainder of this paper, we assume that the directed graph $\mathcal{G}$ is strongly connected, i.e., for each ordered pair of vertices $i, j \in \mathcal{V}$, there is a path from i to j [12].

C. The Ratio-Consensus Algorithm

In this section, we provide an overview of an iterative algorithm—referred to as *ratio consensus*—originally proposed in [13], [14], which serves as a primitive for distributively implementing the frequency regulation and optimal dispatch functions. The ratio-consensus algorithm relies on two linear iterations executed in parallel by the local controller of each DGR, appropriately initialized according to the requirements of the problem.

Consider a small-footprint power system with n DGRs and assume that the communication network describing the exchange of information between them can be described by the graph-theoretic model described above, i.e., a strongly connected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. The local controller of each DGR i participating in the ratio-consensus algorithm maintains two values, y_i and z_i, referred to as internal states, which are (independently) updated at each iteration to be a linear combination of the previous internal states of all nodes in its in-neighborhood. Specifically, for each iteration $k \geq 0$, node i updates its internal states as

$$y_i[k+1] = \sum_{j \in \mathcal{N}_i^-} \frac{1}{\mathcal{D}_j^+} y_j[k], \tag{4}$$

$$z_i[k+1] = \sum_{j \in \mathcal{N}_i^-} \frac{1}{\mathcal{D}_j^+} z_j[k], \tag{5}$$

where $\mathcal{D}_j^+$ is the out-degree of DGR $j \in \mathcal{N}_i^-$. Note that given the self-loop requirement, the update for each internal state includes a weighted value of the previous local internal state of each node. Assuming that $z_i[k] \neq 0, \ \forall k$, at each iteration, the local controller of generator i computes the following:

$$\gamma_i[k] = \frac{y_i[k]}{z_i[k]}. \tag{6}$$

The following lemma, the proof of which can be found in [14], establishes the convergence of (6) to some constant γ, which is equal for every node $i \in \mathcal{V}$.

Lemma 1: Let $y_i[k], \ \forall i$, be the result of iteration (4) for some $y_i[0], \ \forall i$, and $z_i[k], \ \forall i$, be the result of iteration (5) for some $z_i[0], \ \forall i$; then,

$$\lim_{k \to \infty} \gamma_i[k] = \frac{\sum_{j=1}^n y_j[0]}{\sum_{j=1}^n z_j[0]}, \ \forall i.$$

The result in Lemma 1 implies that, through the exchange of information over the directed graph $\mathcal{G}$, ratio consensus allows the DGR local controllers to compute $\sum_{j=1}^n y_j[0] / \sum_{j=1}^n z_j[0]$, thus enabling them to acquire knowledge that is not directly obtainable from their in-neighbors. In the following section, we will show how the ratio-consensus algorithm can be utilized to implement the frequency regulation and optimal dispatch functions in a distributed fashion.

III. Control Objectives and Protocol

First, we provide an overview of the control objectives to be met by our distributed architecture for generation control. These objectives are similar to those sought by centralized architectures for generation control used in bulk power systems, but they are achieved through a distributed computation over the communication network interconnecting the DGRs. This computation relies on two algorithms—their

formal definition is deferred to Section IV—that enable the distributed implementation of the frequency regulation and optimal dispatch functions. We conclude the section by specifying a protocol that allows the DGRs to determine when to utilize each of the two aforementioned algorithms; as we will show, this protocol is based on purely local measurements and as such is completely distributed.

A. Overview of Control Functions and Objectives

Despite being smaller and having lower ratings, the generation control objectives for small-footprint power systems are similar to their larger counterparts. Therefore, the control functions of our distributed architecture are derived from the functionality provided by the typical three-layer generation control architecture used in large power systems. In the discussion that follows, we briefly summarize the control functions of each of these three layers, highlighting the aspects that are relevant to the realization of our distributed architecture.

1) Droop Control: The purpose of this control function is to instantaneously balance generation with demand through local actions taken by the speed governors of each DGR. Typically, the low-level controllers responsible for providing this functionality are designed such that an increase in load corresponding to 100% of the rated output of each DGR results in a 5% drop in frequency, allowing instantaneous load changes to be divided proportionally with respect to DGR ratings [15]. Given that a load increase results in a commensurate decrease in the frequency, primary generation control is often referred to as *droop control*. Regardless of power system size, droop control requires only local measurements and actuations and is therefore inherently decentralized; thus, eliminating the need for a distributed alternative. In the remainder of this paper, we omit further discussion of droop control other than to facilitate discussion of other control functions.

2) Frequency Regulation: While primary generation control provides a simple and effective way to compensate for changes in load, it does so at the expense of frequency deviations. If not eliminated, these variations may result in equipment damage or otherwise undesirable operation, e.g., induction motors and their connected loads rotating at speeds for which they are not designed. In large power systems, there is a control function, commonly referred to as *secondary generation control*—or *automatic generation control* (AGC)—that is responsible for returning the frequency to its nominal value, e.g., 60 Hz, following transients induced by load changes. In addition to regulating frequency, secondary generation control is responsible for maintaining proper interchange power values between control areas. However, in the context of small-footprint power systems, there is no notion of control areas; thus, in our distributed architecture, the only objective of the secondary control is *frequency regulation*.

To compensate for frequency deviations arising from changes in load and the subsequent actions taken by the droop controller of each generator, the secondary generation control in large-scale power systems is commonly implemented using a closed-loop controller that adjusts the set-point of each generator based upon the integral of the frequency error (see, e.g., [10], [15]). The implementation of AGC is centralized with measurements and control signals telemetered to and from the generating units and the control center where the computer implementing the controller resides. In Section!IV-A, we will see that, by relying on a communication network interconnecting the DGRs (as described in Section II-B), when executed over several rounds (properly initialized according to the incremental demand for generation), the ratio-consensus algorithm can be used to replicate the functionality of the aforementioned centralized closed-loop controller used in large power system AGC.

3) Optimal Dispatch: Although primary and secondary generation controls suffice to balance generation with demand and subsequently regulate frequency, they actuate without accounting for operational costs. Consequently, in a large power system, there is a third generation control function which seeks to determine the most economical way to allocate generation demand among all generators. That is, assuming the cost of each generator is a function of its power output, the objective of this tertiary control—commonly referred to as economic dispatch—is to minimize the total generation cost subject to individual capacity limits (see, e.g., [16]).

For large power systems, similar to secondary generation control, the optimal dispatch function relies on a decision-making algorithm executed on a computer that resides in a centralized location, e.g., a control center, with knowledge of each generators' cost function as well as the sum of the power output of all the generators. In Section IV-B, again by relying on a communication network interconnecting the DGRs, we will discuss an algorithm based on ratio consensus that is suitable for distributively implementing the optimal dispatch function.

B. Control Mode Protocol

Large power systems are vast, often containing thousands of loads that vary frequently; consequently, the actions of droop control result in near constant frequency deviations. Furthermore, given that measurements and control signals must be telemetered to and from a central location, secondary and tertiary generation control in bulk power systems do not run continuously, relying instead on time-triggered actuations performed approximately every two-to-five seconds and five minutes, respectively [10].

In the context of small-footprint power systems, each load may be large relative to the total capacity of the DGRs; moreover, the collective inertia of the DGRs may be small. Thus, following a sudden load change, these characteristics may result in large frequency deviations, the dynamics of which are governed by fast time constants. Therefore, while minimizing cost is important, the main control objective in small-footprint power systems is to regulate frequency. As such, rather than operating on a fixed time interval, we adopt an event-triggered control architecture, which only actuates when the frequency error exceeds a specified bound and does not optimally dispatch the DGRs until the frequency is sufficiently close to the nominal value. Then, after returning the frequency to within some bound of the desired value, a distributed algorithm implementing the optimal dispatch function is executed. In the discussion that follows, we specify the protocol utilized by the local controllers to determine the appropriate control mode, and introduce the requisite notation that will be used in subsequent developments.

To return the frequency to its nominal value, following one or more load changes, and subsequently dispatch the DGRs optimally, our architecture relies on the execution of two distributed algorithms (to be described in Section IV) over several rounds. Let r index the rounds over which the distributed algorithms are executed, and denote the generation set-point of DGR i at round r by u_i^r. Then, the relation between the generation set-point of DGR i between two consecutive rounds, r and $r+1$, is given by

$$u_i^{r+1} = u_i^r + \Delta u_i^r, \tag{7}$$

where Δu_i^r is the amount by which DGR i should adjust its output at round r. Define $\omega = \left[\omega_1, \ldots, \omega_n\right]^{\mathrm{T}}$, where ω_i is the electrical angular speed of DGR i, and $u^r = \left[u_1^r, \ldots, u_n^r\right]^{\mathrm{T}}$. Let $\epsilon > 0$ denote the maximum allowable electrical frequency[1] error, and let ξ_i^r be an indicator for the optimal dispatch function for DGR i, i.e.,

$$\xi_i^r = \begin{cases} 0, & \text{if DGR } i \text{ optimally dispatched since last freq. reg.,} \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

Then, the protocol utilized by DGR i to determine the value of Δu_i^r at round r is defined as

$$\begin{aligned} \Delta u_i^r &= h_i(u_i^r, \psi^r) \\ &:= \begin{cases} h_i^f(u_i^r, \psi^r), & \text{if } |\omega_i - \omega_s| > \epsilon, \\ h_i^o(u_i^r, \psi^r), & \text{if } |\omega_i - \omega_s| \leq \epsilon \text{ and } \xi_i^r = 1, \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \tag{9}$$

[1]We assume that, aside from a proportionality constant, the electrical angular speed of DGR i, ω_i, is equivalent to the electrical frequency of the bus to which it is connected, and thus use the terms interchangeably.

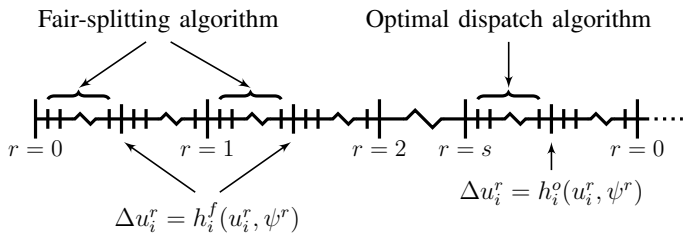


Fig. 2: Timeline of distributed generation control algorithms.

for some function $h_i^f : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ to be defined in Section IV-A, and some function $h_i^0 : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ to be defined in Section IV-B; and where the input ψ^r is given by

$$\psi^r = \begin{cases} \phi^f(\omega^r, u^r), & \text{if } |\omega_i - \omega_s| > \epsilon, \\ \phi^o(u^r), & \text{if } |\omega_i - \omega_s| \leq \epsilon, \end{cases} \tag{10}$$

for some function $\phi^f : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ to be defined in Section IV-A, and some function $\phi^o : \mathbb{R}^n \mapsto \mathbb{R}$ to be defined in Section IV-B.

Given the protocol specified in (8)–(10), the timeline shown in Fig. 2 outlines the relative timescales over which the frequency regulation and optimal dispatch algorithms are executed. If we let s index the round during which the DGRs are optimally dispatched, i.e., the round for which $|\omega_i - \omega_s| \leq \epsilon$ and $\xi_i^r = 1$, the figure illustrates that, for $r < s$, the magnitude of the frequency error exceeds the specified bound and thus the frequency regulation algorithm is executed. In particular, at rounds $r = 0, 1, \ldots, s-1$, the DGRs adjust their generation set-points using $h_i^f(\cdot)$. At $r = s$, the magnitude of the frequency error has been reduced to within the specified bound and the DGRs generation set-points are modified according to $h_i^o(\cdot)$, which yields the optimal generation allocation among the DGRs. Regardless of the algorithm used, after the set-points have been adjusted, the system evolves according to the DAE in (1)-(3).

From (9), we see that in order to determine the incremental set-point of each DGR, one needs the *global* information captured in the value of ψ^r, which is determined by the output of the functions $\phi^f(\cdot)$ and $\phi^o(\cdot)$ in (10). Note also that, in order to evaluate these functions, it is necessary to have the set-points and electrical angular speeds of all the DGRs in the system. Additionally, as we will see in Section IV, $\phi^f(\cdot)$ depends on the minimum and maximum output power of all the DGRs in the system; whereas $\phi^o(\cdot)$ depends on the minimum and maximum output power of all the DGRs in the system, and the parameters of each of the DGR power output cost functions. Although this implies that in order to evaluate (10), and in turn (9), it is necessary to pass all the above information to a centralized processor, in Section IV we will explain how the functions $\phi_i^f(\cdot)$ and $\phi_i^o(\cdot)$ can be evaluated using a distributed computation over the communication network interconnecting the DGR controllers; this distributed computation relies on the ratio-consensus algorithm discussed in Section II-C and obviates the need for a centralized processor.

IV. Control Algorithms

In this section, we describe the two algorithms that serve to distributively compute the functions $h_i^f(\cdot)$ and $h_i^o(\cdot)$ in (9), thus enabling the implementation of frequency regulation and optimal dispatch without the need for a centralized processor.

A. Distributed Frequency Regulation

Consider a small-footprint power system consisting of n DGRs outfitted with local controllers, the interconnections of which can be represented by a strongly connected directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Furthermore, define $\Delta\underline{P}_i^r := u_i^r - \underline{P}_i$, and $\Delta\overline{P}_i^r := \overline{P}_i - u_i^r$ to be the incremental minimum and maximum power output of DGR i at round r, respectively. If we assume that each of the local controllers is capable of obtaining a measurement of the speed of the DGR to which it is connected, an estimate for the amount

by which DGR i should adjust its generation set-point to drive the electrical frequency to the nominal value at round r is

$$\Delta\hat{u}_i^r = \kappa_i^r(\omega_i^r - \omega_s), \tag{11}$$

where ω_i is the electrical angular speed of DGR i, ω_s is the synchronous reference rotating speed, and $\kappa_i^r > 0$ is a gain chosen by the local controller.

With no constraints on power output, a simple solution to the frequency regulation problem is for each DGR to adjust its set-point as in (11). However, if constraints due to DGR power ratings or operational limitations are considered, this solution may not feasible. Thus, in order to address this problem, we use the ratio-consensus algorithm, with appropriately chosen initial conditions that account for DGR limits, to divide the estimated incremental generation demand among all DGRs.

As mentioned previously, the characteristics of the loads and DGRs in small-footprint power systems can lead to large frequency deviations as well as fast transients. Thus, we adopt an allocation scheme for the distributed frequency regulation algorithm whereby the incremental set-point of each DGR is adjusted proportionally to its excess capacity in order to eliminate the frequency error as quickly as possible. Furthermore, to ensure a feasible solution, we assume that, at each round r, the sum of the estimated incremental set-points lies within the collective incremental capacity bounds of the DGRs, i.e.,

$$\sum_{i=1}^{n} \Delta\underline{P}_i^r \leq \sum_{i=1}^{n} \Delta\hat{u}_i^r \leq \sum_{i=1}^{n} \Delta\overline{P}_i^r. \tag{12}$$

Then, the ratio consensus states in (4) and (5) are initialized to $y_i[0] = \Delta\hat{u}_i^r - \Delta\underline{P}_i^r$ and $z_i[0] = \Delta\overline{P}_i^r - \Delta\underline{P}_i^r$, $\forall i \in \mathcal{V}$, respectively. Given these initial conditions, it follows from Lemma 1 that DGR i can asymptotically obtain

$$\begin{aligned}
\psi^r &= \lim_{k\to\infty} \gamma_i^r[k] = \lim_{k\to\infty} \frac{y_i^r[k]}{z_i^r[k]} \\
&= \frac{\sum_{j=1}^n \kappa_j^r(\omega_j^r - \omega_s) - \Delta\underline{P}_j^r}{\sum_{j=1}^n \left(\Delta\overline{P}_j^r - \Delta\underline{P}_j^r\right)} \\
&= \frac{\sum_{j=1}^n \kappa_j^r(\omega_j^r - \omega_s) - (\underline{P}_j^r - u_j^r)}{\sum_{j=1}^n \left(\overline{P}_j^r - 2u_j^r + \underline{P}_j^r\right)} \\
&=: \phi^f(\omega^r, u^r),
\end{aligned} \tag{13}$$

which represents the magnitude of the sum of the estimated set-point adjustments as a percentage of the collective incremental capacity of the DGRs.

After the ratio-consensus algorithm has converged and each DGR has obtained ψ^r, the incremental set-point of DGR i at round r is given as

$$\begin{aligned}
\Delta u_i^r &= \Delta\underline{P}_i^r + \psi^r\left(\Delta\overline{P}_i^r - \Delta\underline{P}_i^r\right) \\
&= u_i^r - \underline{P}_i^r + \psi^r\left(\overline{P}_i^r - 2u_i^r + \underline{P}_i^r\right) \\
&=: h_i^f\left(u_i^r, \psi^r\right),
\end{aligned} \tag{14}$$

and DGR i adjusts its reference command according to (7). Given this choice of initial conditions and the definition of $h_i^f(\cdot)$, we refer to the allocation scheme described in (11)–(14) as *fair-splitting*. Next, we provide an example that illustrates the operation of the fair-splitting scheme. Note that the results for this example as well as the results for Examples 2 and 3 below were obtained using the cyber network of the laboratory testbed that we describe in Section V.

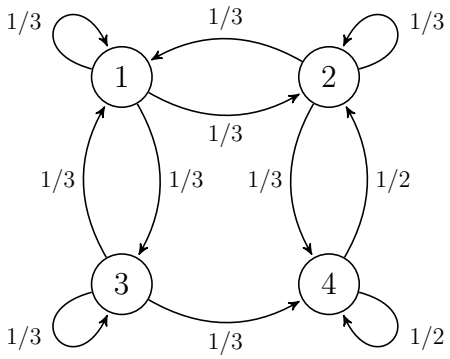


Fig. 3: Graph of 4-node network.

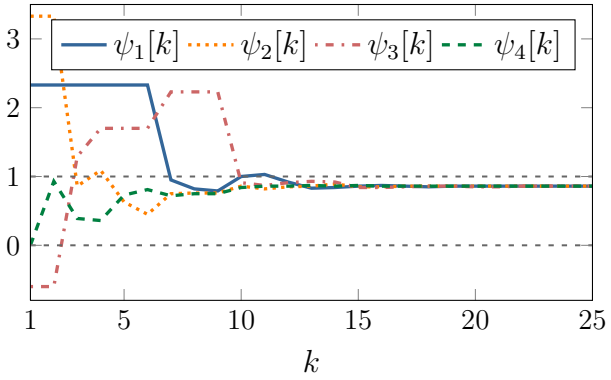


Fig. 4: Fair-splitting algorithm evolution with feasible solution.

Example 1 (Fair-Splitting Feasible Solution): Consider the 4-node network in Fig. 3, where each node represents the local controller of a DGR participating in fair-splitting. Omitting the index of the round, let $\Delta\hat{u} = \left[0.5, 0.5, 0, 0\right]^{\mathrm{T}}$, and the incremental minimum and maximum capacities of the DGRs be given, respectively, by $\Delta\underline{P} = \left[0.15, 0, 0.15, 0.1\right]^{\mathrm{T}}$ and $\Delta\overline{P} = \left[0.3, 0.15, 0.4, 0.25\right]^{\mathrm{T}}$, such that $\sum_{i=1}^{4}\Delta\underline{P}_i \leq \sum_{i=1}^{4}\Delta\hat{u}_i \leq \sum_{i=1}^{4}\Delta\overline{P}_i$. Then, the initial values of y and z are set to $y[0] = \left[0.35, 0.5, -0.15, -0.1\right]^{\mathrm{T}}$, and $z[0] = \left[0.15, 0.15, 0.25, 0.15\right]^{\mathrm{T}}$.

Given the above initial conditions, the evolution of $\psi_i[k],\ i = 1, 2, 3, 4$, over 25 iterations is shown in Fig. 4. From the figure, we see that after approximately 15 iterations, all nodes converge to a solution in which $\psi = 0.86$. Thus, the nodes determine the solution is feasible and compute the incremental generation set-point according to (14), and we have that $\Delta u = \left[0.28, 0.13, 0.36, 0.23\right]^{\mathrm{T}}$. ■

Remark 1: The fair-splitting allocation scheme is not the only viable choice for distributively regulating the frequency using the ratio-consensus algorithm. Other options include allocating incremental set-points based upon: (i) the response time of the DGRs such that the incremental demand is divided proportionally to, for example, the ramp rates of the generation units or the time constant of the governors; or (ii) the proximity of each DGR to its stability limit. We leave investigation of such alternatives to future work.□

If we assign an objective function that takes value zero when the feasible set defined by the inequality constraints in (12) is nonempty or ∞ when the feasible set is empty, the proposed distributed approach to frequency regulation is equivalent to solving a feasibility problem (see, e.g., [17]). In particular, the value of ψ^r found using the fair-splitting procedure can be utilized by each DGR to independently determine if the sum of the incremental set-points is within the collective capacity of the available DGRs. Specifically, at round r, if $\psi^r < 0$ or $\psi^r > 1$, each DGR knows that the collective capacity bounds have been exceeded; we illustrate this idea next.

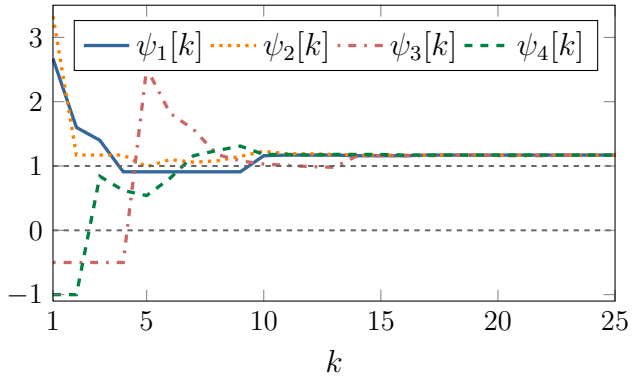


Fig. 5: Fair-splitting algorithm evolution with infeasible solution.

Example 2 (Fair-Splitting Infeasible Solution): Consider the 4-node network in Fig. 3, where each node represents the local controller of a DGR participating in fair-splitting. Omitting the index of the round, let $\Delta\hat{u} = \left[0.5, 0.5, 0, 0\right]^{\mathrm{T}}$, and let the minimum and maximum capacity of the nodes be given respectively as $\Delta\underline{P} = \left[0.1, 0, 0.1, 0.1\right]^{\mathrm{T}}$ and $\Delta\overline{P} = \left[0.25, 0.15, 0.3, 0.25\right]^{\mathrm{T}}$, such that $\sum_{i=1}^{4}\Delta\underline{P}_i = 0.3$ and $\sum_{i=1}^{4}\Delta\overline{P}_i = 0.95 < \sum_{i=1}^{4}\Delta\hat{u}_i$. Then, we have that $y[0] = \left[0.4, 0.5, -0.1, -0.1\right]^{\mathrm{T}}$ and $z[0] = \left[0.15, 0.15, 0.2, 0.1\right]^{\mathrm{T}}$.

The evolution of $\psi_i[k],\ i = 1, 2, 3, 4$, over 25 iterations is shown in Fig. 5. From the figure, we see that after approximately 15 iterations, all nodes converge to a solution in which $\psi = 1.17$. Thus, the nodes determine the solution is infeasible (as they cannot adjust their output beyond their maximum capacities). In Section VI, we show how the global information available through ψ can be used to, e.g., trigger reserve DGRs to come online. ■

B. Distributed Optimal Dispatch

Assuming a feasible solution exists, i.e., (12) is satisfied, the execution of several rounds of the fair-splitting procedure described previously eliminates the frequency error at some round $r = s$. Then, in order to operate the system at the desired frequency, we have that the set-points of the DGRs are such that $\sum_{i=1}^{n} u_i^s := \chi$. As described previously, the fair-splitting procedure adjusts the incremental set-points of the DGRs proportionally with respect to incremental capacity; thus, we would like to reallocate the total generation demand among the DGRs in such a way that minimizes the overall cost of operation. That is, assuming that the cost associated with each DGR is quadratic, we would like to find $u_i^{s*},\ \forall i \in \mathcal{V}$, that satisfy the following constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \frac{\left(u_i^s - \alpha_i\right)^2}{2\beta_i} = \sum_{i=1}^{n} c_i(u_i^s) \\ \text{subject to} \quad & \sum_{i=1}^{n} u_i^s := \chi \\ & 0 \leq \underline{P}_i \leq u_i^s \leq \overline{P}_i,\ \forall i, \end{aligned} \tag{15}$$

where $\alpha_i \leq 0,\ \beta_i > 0$; this problem is commonly referred to as *optimal dispatch* [10].

1) A Centralized Solution to the Optimal Dispatch Problem: The problem of optimally dispatching DGRs at round s given in (15) is convex and has a separable structure [18, pp. 502-506]; thus, its solution can be found by solving the Lagrange dual (see, e.g., [17], [18]), which is given by

$$\begin{aligned} \text{maximize} \quad & \lambda\chi + \sum_{i=1}^{n} f_i(\lambda) \\ \text{subject to} \quad & \lambda \geq 0, \end{aligned} \tag{16}$$

where $f_i(\lambda),\ \forall i$, are

$$f_i(\lambda) = \begin{cases} \frac{(\underline{P}_i-\alpha_i)^2}{2\beta_i} - \lambda\underline{P}_i, & 0 \leq \lambda < \underline{\lambda}_i, \\ -\lambda(\alpha_i + \lambda\frac{\beta_i}{2}), & \underline{\lambda}_i \leq \lambda \leq \overline{\lambda}_i, \\ \frac{(\overline{P}_i-\alpha_i)^2}{2\beta_i} - \lambda\overline{P}_i, & \overline{\lambda}_i < \lambda, \end{cases} \tag{17}$$

with $\underline{\lambda}_i = \frac{\underline{P}_i-\alpha_i}{\beta_i} > 0$ and $\overline{\lambda}_i = \frac{\overline{P}_i-\alpha_i}{\beta_i} > 0$.

The Lagrange dual problem defined by (16) is convex and, in this case, provides the optimal solution to the primal problem. Furthermore, the cost function in (17) is continuously differentiable, i.e., $f_i(\cdot) \in \mathcal{C}^1,\ \forall i$; thus, if the dual problem is feasible, the optimal solution λ^* must satisfy

$$\chi - \sum_{i=1}^{n} g_i(\lambda^*) = 0, \tag{18}$$

where $g_i(\lambda) = -\frac{df_i(\lambda)}{d\lambda}$, i.e.,

$$g_i(\lambda) = \begin{cases} \underline{P}_i, & 0 \leq \lambda < \underline{\lambda}_i, \\ \alpha_i + \lambda\beta_i, & \underline{\lambda}_i \leq \lambda \leq \overline{\lambda}_i, \\ \overline{P}_i, & \overline{\lambda}_i < \lambda. \end{cases} \tag{19}$$

Now, obtaining the value λ^* that satisfies (18) is equivalent to finding the intersection point of the functions $g(\lambda) := \sum_{i=1}^{n} g_i(\lambda)$ and $h(\lambda) := \chi$; this can be accomplished by evaluating the function $g(\cdot)$ for, at most, $2n$ values corresponding to the elements in $\mathcal{U} := \left\{\underline{\lambda}_1, \overline{\lambda}_2, \ldots, \underline{\lambda}_n, \overline{\lambda}_n\right\}$. To this end, define $\mathcal{U}^+ := \{\lambda \in \mathcal{U} : g(\lambda) \geq \chi\}$, and $\mathcal{U}^- := \{\lambda \in \mathcal{U} : g(\lambda) \leq \chi\}$; then, as proposed in [19], the value of λ^* is given by

$$\begin{aligned} \lambda^* &= \lambda^+ - \left(g(\lambda^+) - \chi\right) \frac{\lambda^+ - \lambda^-}{g(\lambda^+) - g(\lambda^-)} \\ &=: \phi^o(u^s), \end{aligned} \tag{20}$$

where

$$\lambda^+ = \underset{\lambda\in\mathcal{U}^+}{\operatorname{argmin}} \, |g(\lambda) - \chi|, \tag{21}$$

$$\lambda^- = \underset{\lambda\in\mathcal{U}^-}{\operatorname{argmin}} \, |g(\lambda) - \chi|, \tag{22}$$

and the solution to the primal problem given in (15) is

$$u_i^{s*} = g_i(\lambda^*),\ \forall i. \tag{23}$$

From (20), it is easy to see that λ^* is obtained by a linear interpolation between the values of λ^- and λ^+; for this reason, we refer to the procedure outlined in (20)–(23) as the *interpolation method*.

2) A Distributed Implementation of the Interpolation Method: Although (23) provides the unique global solution to the optimal dispatch problem by solving its Lagrange dual, in order to determine $g(\lambda^+)$ and $g(\lambda^-)$, a centralized entity with knowledge of all the individual $g_i(\lambda)$'s and χ is required. If we rearrange (20), however, we will see that ratio consensus, together with a simple message-passing protocol, can be utilized to find λ^* without requiring a centralized decision maker or the need for each DGR to obtain all the individual functions $g_i(\cdot)$, the values they take for λ^+ or λ^-, or the value of χ.

First, note that (20) can be rewritten as

$$\lambda^* = \lambda^+ - \left(\frac{g(\lambda^+)}{\chi} - 1\right) \frac{\lambda^+ - \lambda^-}{\frac{g(\lambda^+)}{\chi} - \frac{g(\lambda^-)}{\chi}}, \tag{24}$$

where

$$\frac{g(\lambda^+)}{\chi} = \frac{\sum_{i=1}^n g_i(\lambda^+)}{\sum_{i=1}^n u_i^s}, \tag{25}$$

$$\frac{g(\lambda^-)}{\chi} = \frac{\sum_{i=1}^n g_i(\lambda^-)}{\sum_{i=1}^n u_i^s}, \tag{26}$$

which are ratios of sums of values that are known or can be computed by each local controller. As (25) and (26) imply, however, to find the optimal solution, each DGR must know λ^+ and λ^- *a priori*, that is, if each node knew which two $\lambda \in \mathcal{U}$ satisfied (21) and (22), ratio consensus could be used by the nodes to asymptotically obtain the ratios defined in (25) and (26), and consequently λ^*. From (24), we see that the criteria for determining λ^+ and λ^- are equivalent to finding $\lambda \in \mathcal{U}$ such that $g(\lambda)/\chi$ is closest to 1 from above and below, respectively; thus, if each DGR obtains $g(\lambda)/\chi$, $\forall \lambda \in \mathcal{U}$, it can determine which ratios correspond to $g(\lambda^+)/\chi$ and $g(\lambda^-)/\chi$. Furthermore, if each DGR also knows which $\lambda \in \mathcal{U}$ correspond to λ^+ and λ^-, λ^* can be computed and the optimal solution to the primal problem can be found using (23). Next, we provide a three-step procedure that enables the nodes to distributively perform the tasks above; the steps of this procedure are as follows.

[S1.] Without loss of generality, assume that initially, each node i only knows its respective $\underline{\lambda}_i$ and $\overline{\lambda}_i$. Thus, by broadcasting these values, all nodes in the out-neighborhood of i can obtain them, and in turn, broadcast them to their out-neighbors. Then, after a finite number of steps, bounded by the diameter of the graph representing the communication network, each node i will obtain every $\lambda \in \mathcal{U}$.

[S2.] Once each node has acquired every $\lambda \in \mathcal{U}$, ratio consensus is used with $2n$ numerator states and a single denominator state. Let $\underline{y}_i^{(j)}$ and $\overline{y}_i^{(j)}$, for $j = 1, \ldots, n$, and z_i, be the numerator and denominator states maintained by node i, respectively. Then, if the states are initialized such that $\underline{y}_i^{(j)}[0] = g_i(\underline{\lambda}_j)$, $\overline{y}_i^{(j)}[0] = g_i(\overline{\lambda}_j)$, for $j = 1, \ldots, n$, and $z_i[0] = u_i^s$, it follows from Lemma 1 that each node i can asymptotically obtain

$$\underline{\gamma}_i^{(j)} = \lim_{k\to\infty} \frac{\underline{y}_i^{(j)}[k]}{z_i[k]} = \frac{\sum_{l=1}^n g_l(\underline{\lambda}_j)}{\sum_{l=1}^n u_l^s} = \frac{g(\underline{\lambda}_j)}{\chi}, \tag{27}$$

$$\overline{\gamma}_i^{(j)} = \lim_{k\to\infty} \frac{\overline{y}_i^{(j)}[k]}{z_i[k]} = \frac{\sum_{l=1}^n g_l(\overline{\lambda}_j)}{\sum_{l=1}^n u_l^s} = \frac{g(\overline{\lambda}_j)}{\chi}, \tag{28}$$

for $j = 1, \ldots, n$, which correspond to $g(\lambda)/\chi$, $\forall \lambda \in \mathcal{U}$. Therefore, by determining which ratios are closest to 1 from above and below and the corresponding λ^+ and λ^-, each DGR can compute λ^* using (24).

[S3.] After the values $\underline{\gamma}_i^{(j)}$ and $\overline{\gamma}_i^{(j)}$, $i, j = 1, \ldots, n$, have converged according to the ratio-consensus algorithm, and each node has determined λ^*, by letting $\psi^s = \lambda^*$, the incremental set-point of DGR i at round $r = s$ is given by

$$\begin{aligned} \Delta u_i^s &= g_i(\psi^s) - u_i^s \\ &= \begin{cases} \underline{P}_i - u_i^s, & 0 \le \psi^s < \underline{\lambda}_i, \\ \alpha_i + \psi^s \beta_i - u_i^s, & \underline{\lambda}_i \le \psi^s \le \overline{\lambda}_i, \\ \overline{P}_i - u_i^s, & \overline{\lambda}_i < \psi^s. \end{cases} \\ &=: h_i^o(u_i^s, \psi^s), \end{aligned} \tag{29}$$

and DGR i adjusts its reference command according to (7).

Algorithm 1: Distributed optimal dispatch

Each node $i,\ i = 1, \ldots, n,$ **separately does the following:**
Input: $\alpha_i, \beta_i, \underline{P}_i, \overline{P}_i, u_i^s$
Output: Δu_i^s

begin

initialize states:

$$\underline{\lambda}_i = \frac{\underline{P}_i - \alpha_i}{\beta_i} \qquad \overline{\lambda}_i = \frac{\overline{P}_i - \alpha_i}{\beta_i}$$
$$\underline{y}_i^{(i)}[0] = g_i(\underline{\lambda}_i) \qquad \overline{y}_i^{(i)}[0] = g_i(\overline{\lambda}_i)$$
$$\underline{y}_i^{(j)}[0] = 0 \qquad \overline{y}_i^{(j)}[0] = 0,\ \forall j \neq i$$
$$z_i[0] = u_i^s$$

foreach *iteration,* k **do**

if $\underline{\lambda}_i$, $\overline{\lambda}_i$ *not broadcasted* **then**

broadcast $\underline{\lambda}_i$, $\overline{\lambda}_i$

if *received* $\underline{\lambda}_j$, $\overline{\lambda}_j$ **then**

$$\underline{y}_i^{(j)}[k] = \underline{y}_i^{(j)}[k] + g_i(\underline{\lambda}_j)$$
$$\overline{y}_i^{(j)}[k] = \overline{y}_i^{(j)}[k] + g_i(\overline{\lambda}_j)$$

update and broadcast states:

$$\underline{y}_i^{(j)}[k+1] = \sum_{l \in \mathcal{N}_i^-} \underline{y}_l^{(j)}[k]$$
$$\overline{y}_i^{(j)}[k+1] = \sum_{l \in \mathcal{N}_i^-} \overline{y}_l^{(j)}[k]$$
$$z_i[k+1] = \sum_{l \in \mathcal{N}_i^-} z_l[k]$$

compute:

$$\underline{\gamma}_i^{(j)} = \frac{\underline{y}_i^{(j)}[\infty]}{z_i[\infty]} \qquad \overline{\gamma}_i^{(j)} = \frac{\overline{y}_i^{(j)}[\infty]}{z_i[\infty]}$$

determine $g(\lambda^+)/\chi$, $g(\lambda^-)/\chi$, λ^+, λ^-
compute $\psi^s = \lambda^*$ according to (24)
return $\Delta u_i^s = g_i(\psi^s) - u_i^{s-1}$

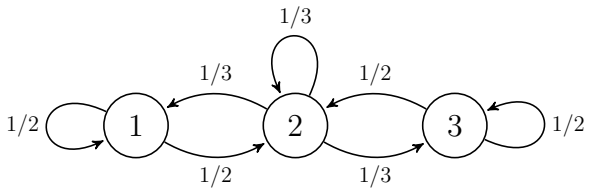


Fig. 6: Graph of linear 3-node network.

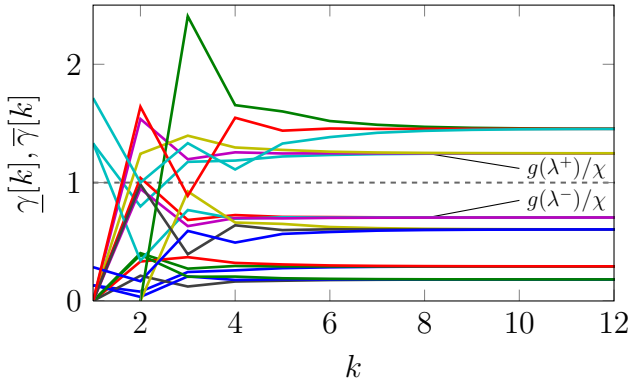


Fig. 7: Example evolution of the ratio-consensus algorithm.

Although the three-step procedure described above implies that each DGR must obtain every $\lambda \in \mathcal{U}$ (Step **S1**) before proceeding to run the ratio-consensus algorithm (Step **S2**), it is possible, with a slight modification, to execute the message-passing protocol and ratio consensus in parallel and still guarantee convergence of (27) and (28) (a formal proof can be found in [20]). In particular, each node i initializes z_i as before, but only node j initializes $\underline{y}_j^{(j)}[0] = g_j(\underline{\lambda}_j)$ and $\overline{y}_j^{(j)}[0] = g_j(\overline{\lambda}_j)$ since it is the only node that possesses $\underline{\lambda}_j$ and $\overline{\lambda}_j$ at $k = 0$. The remaining nodes initialize their numerator states corresponding to DGR j to zero, i.e., $\underline{y}_i^{(j)}[0] = 0$ and $\overline{y}_i^{(j)}[0] = 0$, for $i \neq j$, and, upon receiving $\underline{\lambda}_j$ and $\overline{\lambda}_j$, calculate $g_i(\underline{\lambda}_j)$ and $g_i(\overline{\lambda}_j)$ and add it to $\underline{y}_i^{(j)}$ and $\overline{y}_i^{(j)}$, respectively. The three-step procedure for distributively solving the optimal dispatch problem, including the modification to allow the message-passing protocol and ratio consensus to run in parallel, is described in Algorithm 1; its operation is illustrated in the following example.

Example 3 (Optimal Dispatch): Consider the graph in Fig. 6 which represents the exchange of information between DGR local controllers in a 3-node network. We demonstrate the distributed optimal dispatch algorithm by showing the evolution of the γ's and $\overline{\gamma}$'s as computed by each of the local controllers. Let $\alpha = \big[-13/6, -3/2, -15/8\big]$, $\beta = \big[1/6, 1/2, 1/8\big]$, $\underline{P} = \big[10, 10, 20\big]$, and $\overline{P} = \big[100, 100, 120\big]$. Also, omitting the index of the round, let $u = \big[75, 75, 70\big]$ such that $\chi = 220$. Given the above values of α, β, $\underline{P}$, and $\overline{P}$, and the definitions given in Section IV-B, we have that $\underline{\lambda} = \big[73, 23, 175\big]$ and $\overline{\lambda} = \big[613, 203, 975\big]$.

The evolution of $\underline{\gamma}_i^{(j)}$ and $\overline{\gamma}_i^{(j)}$, for $i, j = 1, 2, 3$, is shown in Fig. 7. From the figure, we see that all of the values converge after around 9 iterations, allowing the nodes to determine which ratios are closest from below and above to 1, i.e., $g(\lambda^-)/\chi$ and $g(\lambda^-)/\chi$, the corresponding values of λ^- and λ^+, and the value of λ^* according to (24). After determining $\psi = \lambda^* = 425.29$, each node computes the amount by which it should adjust its set-point according to (29) and we have that $\Delta u = \big[-6.29, 25, -18.71\big]^{\mathrm{T}}$. ■

Remark 2: The message-passing protocol described in Step **S1** is not the only method by which each node can obtain every $\lambda \in \mathcal{U}$. In fact, it is possible to use ratio consensus with $3n$ states to disseminate the necessary knowledge. To use the ratio-consensus algorithm, each node maintains $2n$ numerator states, corresponding to each $\lambda \in \mathcal{U}$, and an additional n denominator states. Let $\underline{v}_i^{(j)}$ and $\overline{v}_i^{(j)}$ be the numerator states and $w_i^{(j)}$ be the denominator states, for $j = 1, \ldots, n$, maintained by node i. Then, if the states

are initialized such that $\underline{v}_i^{(j)}[0] = \underline{\lambda}_i$, $\overline{v}_i^{(j)}[0] = \overline{\lambda}_i$, and $w_i^{(j)}[0] = 1$ if $j = i$, and $\underline{v}_i^{(j)}[0] = \overline{v}_i^{(j)}[0] = w_i^{(j)}[0] = 0, \forall j \neq i$, it follows from Lemma 1 that $\underline{\pi}_i^{(j)} = \lim_{k\to\infty} \frac{\underline{v}_i^{(j)}[k]}{w_i^{(j)}[k]} = \frac{\sum_{l=1}^n \underline{v}_l^{(j)}[0]}{\sum_{l=1}^n w_l^{(j)}[0]} = \frac{\underline{\lambda}_i}{1}$, and $\overline{\pi}_i^{(j)} = \lim_{k\to\infty} \frac{\overline{v}_i^{(j)}[k]}{w_i^{(j)}[k]} = \frac{\sum_{l=1}^n \overline{v}_l^{(j)}[0]}{\sum_{l=1}^n w_l^{(j)}[0]} = \frac{\overline{\lambda}_i}{1}$, for $i = 1, \ldots, n$. □

V. Laboratory Testbed

We begin this section by describing the configuration and hardware specifications of the electrical network of a microgrid built in a laboratory to illustrate the ability of the distributed control architecture to control a small-footprint power system. We first provide the specifications for the hardware used to implement the microgrid's cyber network for communication, computation, and control. Then, we discuss the communication protocol created to exchange information for the distributed algorithms as well as a modification to ratio consensus which increases its robustness. Finally, we give an overview of a protocol used to synchronize the clocks of the nodes in the cyber network; this is important in order to ensure the correct execution of the control algorithms.

A. Electrical Network Hardware

To demonstrate the ability of the proposed distributed architecture to control a set of DGRs, we constructed the six-bus, 240 V, 3-phase power system shown in Fig. 8. The system is comprised of 3 Hampden Engineering synchronous machines, G_1, G_2, and G_3, and 3 wye-connected resistive loads, labeled as P_1, P_2, and P_3. Each synchronous machine is connected at the shaft to a Kollmorgen Goldline brushless permanent magnet synchronous servomotor which serves as the prime mover. The synchronous machines have 3 pole-pairs, therefore to maintain an electrical frequency of 60 Hz, the mechanical speed of the generators is regulated to 1200 rpm. The per-phase resistance of each load is adjusted by adding 500 Ω resistors in parallel. Each load can have up to 10 resistors in parallel per phase, yielding resistances in the range $500, 250, \ldots, 50\,\Omega$. Extra impedance is added via series inductors between bus 4 and bus 6 as well as between bus 1 and bus 3 as shown in Fig. 8. The per-phase inductances, the synchronous generator, and prime mover data are given in Appendix A.

B. Cyber Network Hardware

The hardware chosen to create the cyber network for communication and computation is based upon the open-source electronics prototyping platform Arduino. While there are several other suitable hardware choices, we chose Arduino for its ease of use and for the extensive software libraries and extension circuit boards, called shields, that are available [21].

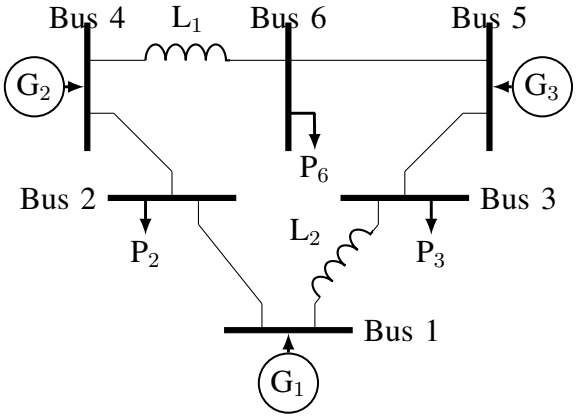
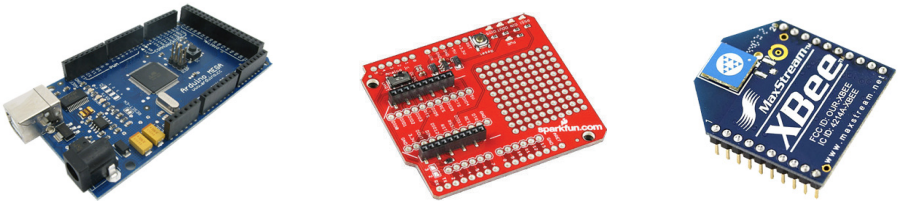


Fig. 8: One-line diagram of small-footprint power system.



(a) Arduino Mega 2560 (b) SparkFun XBee Shield (c) MaxStream XBee Module

Fig. 9: Testbed hardware

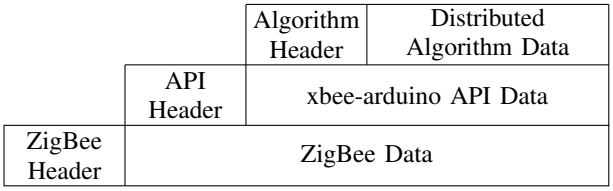


Fig. 10: Communication protocol stack.

Each node in the cyber network is comprised of an Arduino Mega 2560 microcontroller board [22], connected to a MaxStream XB24-DMCIT-250 revB XBee module [23] via a SparkFun Electronics XBee shield [24]. The Arduino Mega 2560, shown in Fig. 9a, is based on the Atmel AVR ATmega2560 [25], an 8-bit microcontroller with 256 kB of flash memory, a clock speed of 16 MHz, and four universal asynchronous receiver/transmitter (UART) ports. The XBee shield, shown in Fig. 9b, serves as an interface between the Arduino board and the XBee module while providing the requisite 3.3 V power supply via a voltage regulator. The XBee, shown in Fig. 9c, is an embedded RF module with a built-in chip antenna operating at 2.4 GHz that allows the nodes in the testbed to exchange packetized data wirelessly.

Although the XBee shield allows for plug-and-play operation between the Arduino and the XBee, it is configured to utilize the same UART port as the onboard USB-to-serial converter, preventing the Arduino from simultaneously communicating with a computer via the USB port and other nodes in the testbed via XBee. Thus, to allow for concurrent connections, we use a modified shield which routes the XBee UART to an alternative port on the Arduino. While this modification is useful for logging the evolution of the distributed algorithms, it is also necessary for enabling the Arduinos to control the prime movers connected to the synchronous generators in the electrical network.

C. Cyber Network Software

Next, we discuss the communication protocol developed to exchange information among nodes in the cyber network, as well as a modification to the ratio consensus algorithm that enables robust operation in the presence of unreliable communication links. Additionally, we provide an overview of the protocol used to synchronize the clocks of the cyber network nodes.

1) Communication Protocol: The communication protocol created to exchange information among the cyber network nodes can be described by the three abstraction layers illustrated in Fig. 10. The lowest layer, commonly referred to as medium access control (MAC), is implemented on the XBee modules and is based upon the ZigBee (IEEE 802.15.4) standard [26]. The middle layer is a version of the xbee-arduino API [27], modified to account for the aforementioned alteration to the XBee shield and to enable incoming and outgoing packets to be time-stamped in order to increase the accuracy of the synchronization mechanism discussed later in this section. Each packet generated by the xbee-arduino API contains information about the sender and the intended recipient, allowing the Arduinos to send unicast messages as well as determine the source of broadcasted packets. Note that to enable use of the xbee-arduino API, the XBee modules are placed in API mode (AP=2 with escapes). The header of the top layer contains information about which algorithm is being used, i.e., fair splitting or optimal dispatch, while the payload contains the actual information being exchanged.

In order to take advantage of the wireless medium used for communication among nodes, all of the packets used to exchange values for the distributed algorithms are broadcasted; that is, packets are not addressed to a particular node. Furthermore, to minimize network traffic, no acknowledgements are sent upon successful receipt of packets. To create a partially connected network despite the close proximity of the Arduinos during experimentation, each device is programmed to only accept packets received from the nodes in its in-neighborhood. In a more realistic setup, however, the testbed could be adapted to allow the availability of links between nodes to be determined dynamically based upon, for example, signal strength.

Algorithm 2: Robust ratio-consensus algorithm

Input: $y_i[0]$, $\mu_i[0]=0$, $\nu_{ij}[0]=0$, $\forall j \in \mathcal{N}_i^-$
$z_i[0]$, $\sigma_i[0]=0$, $\tau_{ij}[0]=0$, $\forall j \in \mathcal{N}_i^-$

for $k \geq 0$:

Compute:

$\mu_i[k+1] = \mu_i[k] + y_i[k]/\mathcal{D}_i^+$

$\sigma_i[k+1] = \sigma_i[k] + z_i[k]/\mathcal{D}_i^+$

Broadcast: $\mu_i[k+1]$ and $\sigma_i[k+1]$ to all $l \in \mathcal{N}_i^+,\ l \neq i$

Receive: $\mu_j[k+1]$ and $\sigma_j[k+1]$ from each $j \in \mathcal{N}_i^-$ if $(i,j) \in \mathcal{E}[k]$

Compute:

$$\eta_{ij}[k+1] = \begin{cases} \mu_j[k+1]\,, & \text{if } (i,j) \in \mathcal{E}[k], \\ \nu_{ij}[k]\,, & \text{if } (i,j) \notin \mathcal{E}[k]. \end{cases}$$

$$y_i[k+1] = \tfrac{1}{\mathcal{D}_i^+} y_i[k] + \sum_{\substack{j \in \mathcal{N}_i^- \\ i \neq j}} (\nu_{ij}[k+1] - \nu_{ij}[k])$$

$$\eta_{ij}[k+1] = \begin{cases} \sigma_i[k+1]\,, & \text{if } (i,j) \in \mathcal{E}[k], \\ \eta_{ij}[k]\,, & \text{if } (i,j) \notin \mathcal{E}[k]. \end{cases}$$

$$z_i[k+1] = \tfrac{1}{\mathcal{D}_i^+} z_i[k] + \sum_{\substack{j \in \mathcal{N}_i^- \\ i \neq j}} (\tau_{ij}[k+1] - \tau_{ij}[k])$$

Output: $\gamma_i[k+1] = y_i[k+1]/z_i[k+1]$

2) Ratio Consensus Implementation: Although the bottom layer of the protocol stack is designed to minimize packet collisions, it is still possible for packet losses to occur, particularly when attempting to reduce the time required to compute new generation commands using the distributed algorithms. To mitigate the effects of information lost due to temporarily unavailable communication links resulting from, e.g., neighboring nodes attempting to transmit simultaneously, we use a modified version of the ratio-consensus algorithm originally proposed in [28]. We refer to this variant of the ratio-consensus algorithm as *robust ratio consensus*; its pseudocode[2] is provided in Algorithm 2. In Appendix B, we provide a brief overview of the operation of Algorithm 2 and discuss the implications with respect to its implementation in the laboratory testbed. [In [28], it was shown that, under certain probabilistic assumptions on the link availability model, the asymptotic value of $\gamma_i[k]$ obtained with Algorithm 2 is identical to the original ratio-consensus algorithm described in Section II-C with probability one.]

[2]In order to take into account for the possibility that communication links may not be available at every iteration, it is necessary to slightly modify the graph-theoretic model describing the exchange of information among DGRs introduced earlier. To this end, in Algorithm 2, we denote the graph representing the network interconnecting the DGRs as $\mathcal{G}[k] = \{\mathcal{V}, \mathcal{E}[k]\}$, where $\mathcal{V}$ is independent of k as defined before, and $\mathcal{E}[k]$ is the set of edges where $(i,j) \in \mathcal{E}[k]$ if DGR i can receive information from DGR j at iteration k. We assume that $\mathcal{E}[k] \subseteq \mathcal{E}$, $\forall k \geq 0$, where $\mathcal{E}$ is as defined in Section II-B, and describes the scenario in which all communication links are available.

3) Synchronization Protocol: Throughout the formulation of the ratio-consensus algorithm and its robust variant, we assumed that the local controller of all participating DGRs update the value of their state variables in unison; i.e., node i updates its state at iteration k at the same time node j updates its state, $\forall i, j, \in \mathcal{V}$. Without a common time reference and with no acknowledgements, however, it is possible for the local controllers to update their states at different times which may result in convergence to an inaccurate solution. While robust ratio-consensus reduces the sensitivity of the system to timing errors, it is still possible for the nodes to converge to the wrong solution. Thus, before the distributed control algorithm is executed, all nodes are synchronized to a common time reference. The synchronization mechanism used in our hardware testbed is based on the hierarchy referencing time synchronization (HRTS) protocol proposed in [29]. This protocol has low overhead requirements and is capable of synchronizing the clocks of several nodes to a reference using only three packets. A detailed description of the synchronization process is provided In Appendix C.

VI. Experimental Results

Using the laboratory testbed described in the previous section, we verify the effectiveness of our distributed generation control architecture under a variety of scenarios. We first present results demonstrating the distributed frequency regulation and optimal dispatch functions controlling the DGRs in the small-footprint power system, following both an increase and a decrease in load. We then add a fourth DGR to the system which acts as a spinning reserve to illustrate how the local controller of each DGR can independently determine if the collective capacity has been exceeded and act accordingly.

Throughout this section, the limits and set-points of the prime movers are given as torque rather than as power. Also, instead of having each DGR estimate the amount by which its set-point should be adjusted according to (11), in the results that follow, we select a single DGR, referred to as the leader, which is responsible for estimating the total torque that needs to be added or removed at each round. In the remainder of this section, the coefficients defining the DGR cost functions are $\alpha = \left[-13/6, -3/2, -15/8\right]^{\mathrm{T}}$ N·m and $\beta = \left[1/6, 1/2, 1/8\right]^{\mathrm{T}}$ N·m. Additionally, the bound used by the leader to determine when to trigger the optimal dispatch algorithm as defined in (9) is $\epsilon = 8$ rpm. As noted in the previous section, the synchronous generators have 3 pole pairs; thus, in order to regulate the frequency to 60 Hz, the machines should have a mechanical speed of $\omega_s = 1200$ rpm.

A. Load Increase

We begin by illustrating the system response to a load increase. In this experiment, the exchange of information between local DGR controllers is represented by the graph shown in Fig. 6, where nodes $1, 2, 3$ correspond to generators G_1, G_2, G_3 in Fig. 8, respectively. The minimum output torque of all DGRs is 0 N·m while the maximum output torques for DGRs G_1, G_2, and G_3 are 4, 2.5, and 3.5 N·m, respectively. As mentioned in the previous section, one of the nodes must broadcast a scheduling packet specifying the start time, number of iterations, and period of each iteration. [We make node 1 the leader and set the gain it uses to compute Δu according to (11) to $\kappa_1^r = 0.0017$ N·m·s/rad for all r.] At each round r, node 1 broadcasts a scheduling packet, specifying 50 iterations with a period of 50 ms for the frequency regulation algorithm, and 100 iterations with 300 ms periods for the optimal dispatch algorithm.

The plot in Fig. 11 shows the torque set-points for each of the three generators as well as the speed of DGR 1 as the system responds to two load increases at times $t = 100$ s and $t = 400$ s, and a subsequent generator re-dispatch at time $t = 775$ s which is triggered to minimize the overall generation cost. For the first and second load changes, the total power drawn is increased from 313 W to 396 W, and from 396 W to 465 W by adjusting the resistive load from 500 Ω/phase to 250 Ω/phase at buses 3 and 2, respectively.

From Fig. 11 we see that, following the first load increase, the speed of DGR 1 returns to 1200 rpm, i.e., system frequency returns to 60 Hz, after four rounds of the fair-splitting algorithm. If we let $r = 0$ be the index of the round before the first load increase and subsequent execution of the frequency regulation algorithm, then the value of ψ^r as defined in (13) for $r = 1, 2, 3, 4$ is 0.58, 0.59, 0.59, and 0.60, respectively.

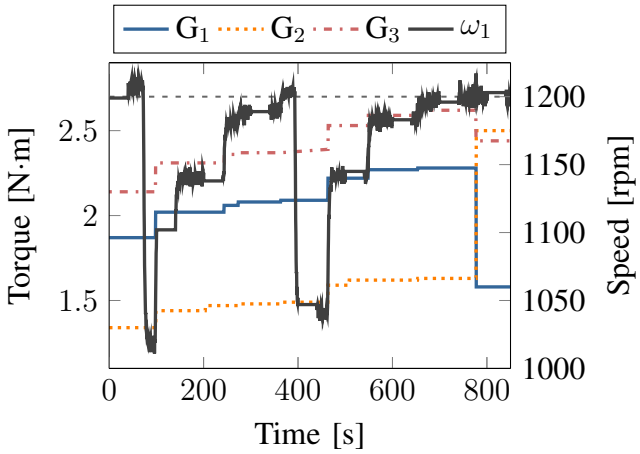


Fig. 11: System response to load increase.

Similar to the first load change, three rounds of the fair-splitting algorithm, corresponding to $r = 5, 6, 7$, return the frequency to the nominal value following the second load change. The value of ψ^r for the last three rounds of the frequency regulation algorithm are 0.63, 0.65, and 0.65, respectively.

After the two load changes and the execution of 7 rounds of the fair-splitting algorithm, the frequency is within the specified bound, i.e., $|\omega_1 - \omega_s| \leq \epsilon$, and, as described by (9), the optimal dispatch function is used to determine the amount by which the DGRs should adjust their output in order to minimize the overall cost. If we let $s = 8$ be the index of the round that the DGRs are re-dispatched optimally, then the local controllers determine that $\psi^s = 276.36$ N·m, and, as Fig. 11 shows, the DGRs adjust their set-points according to (29) at time $t = 775$ s.

B. Load Decrease

We now illustrate the system response to a load decrease. For this experiment, the cyclic graph in Fig. 12 represents the exchange of information among the DGR local controllers. Similar to the previous experiment, node 1 is selected to be the leader with $\kappa_1^r = 0.0018$ N·m·s/rad for all r, the minimum output torques of all DGRs are 0 N·m, and the maximum output torques for DGRs $1, 2$, and 3 are 4, 2.5, and 3.5 N·m, respectively. As before, the leader broadcasts a scheduling packet to the nodes, specifying 75 and 125 iterations and 50 ms and 300 ms periods for the frequency regulation and optimal dispatch algorithms, respectively.

The plot in Fig. 13 shows the torque set-points for each of the three generators as well as the speed of DGR G_1 as the system responds to two load decreases at times $t = 100$ s and $t = 300$ s, and a subsequent generator re-dispatch at time $t = 500$ s, which is triggered to minimize the overall generation cost. For the first and second load changes, the total power drawn is decreased from 390 W to 365 W, and from 365 W to 352 W by adjusting the resistive load at bus 6 from 500 Ω/phase to 750 Ω/phase, and from 750 Ω/phase to 1000 Ω/phase, respectively.

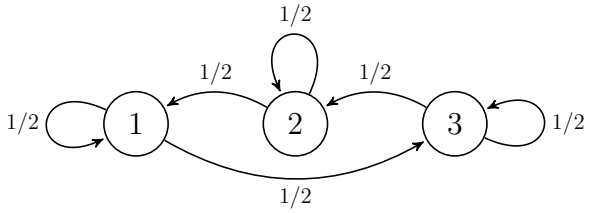


Fig. 12: Graph of cyclic 3-node network.

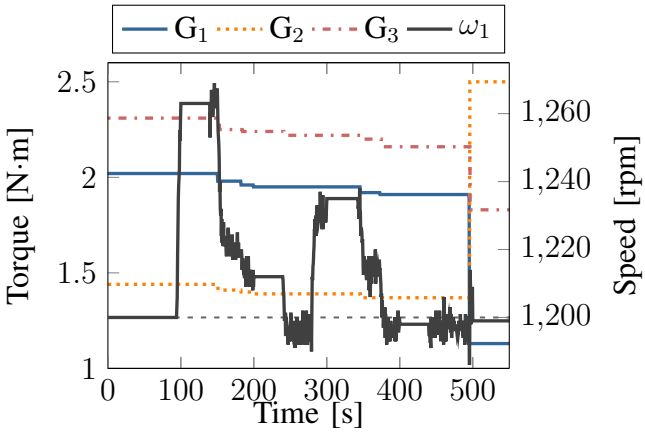


Fig. 13: System response to load decrease.

Following the first load decrease, we see from Fig. 13 that the speed of DGR G_1 returns to 1200 rpm after three rounds of the fair-splitting algorithm. If we let $r = 0$ be the index of the round before the first load change and subsequent execution of the frequency regulation algorithm, then the value of ψ^r for $r = 1, 2, 3$ is 0.56, 0.56, and 0.56, respectively. After the second load change, two rounds of the fair-splitting algorithm, corresponding to $r = 4$ an 5, return the frequency to the nominal value. The value of ψ^r for the last two rounds of the frequency regulation algorithm is 0.55 and 0.54.

After the two load changes and the execution of 5 rounds of the fair-splitting algorithm, the frequency is within the specified bound and the generators are re-dispatched in order to minimize the overall cost. If we let $s = 6$ be the index of the round that the optimal dispatch function is executed, then the local controllers determine that $\psi^s = 243.57$ N·m and the DGRs adjust their set-points according to (29) at time $t = 500$ s as shown in Fig. 13.

C. Load Increase with Spinning Reserve

As illustrated in Example 2, the value of ψ^r obtained through the execution of the function for distributed frequency allows each DGR local controller to independently determine if the collective capacity has been exceeded. Next, we demonstrate how this information can be used by a DGR acting as a spinning reserve to come online.

In order to demonstrate the capability of each generator to independently determine when the demand for generation exceeds the collective generating capacity of the DGRs online, we connected an additional DGR to bus 2, which we refer to as DGR G_4. Initially, the additional DGR limits its output to the minimum torque required to operate at synchronous speed, but participates in the distributed algorithm with a maximum torque of 0.95 N·m. The actual maximum torque of the DGR G_4 is 2 N·m, while DGRs G_1, G_2, and G_3 have maximum constraints of 2.5 N·m, 2.25 N·m, and 1.75 N·m, respectively; the minimum torque is 0.1 N·m for all DGRs. The exchange of information among the DGRs conforms to the graph in Fig. 3.

The plot in Fig. 14 shows the prime mover torque and the average mechanical speed as the electrical load is increased. At $t \approx 275$ s, DGRs G_1, G_2, and G_3 reach their maximum power output and thus cannot increase their output to regulate the frequency. At this point, by obtaining a value of ψ^r greater than one, DGR G_4 determines that the demand exceeds the collective capacity and adjusts its maximum torque limit to its actual value of 2 N·m. This occurs at $t \approx 360$ s, after which all DGRs are able to increase their speed to the nominal value of $\omega_s = 1200$ rpm. Once the reserve generator is switched in and the frequency is returned to the nominal value, the optimal dispatch function is executed and the DGRs adjust their set-points at time $t \approx 650$ s.

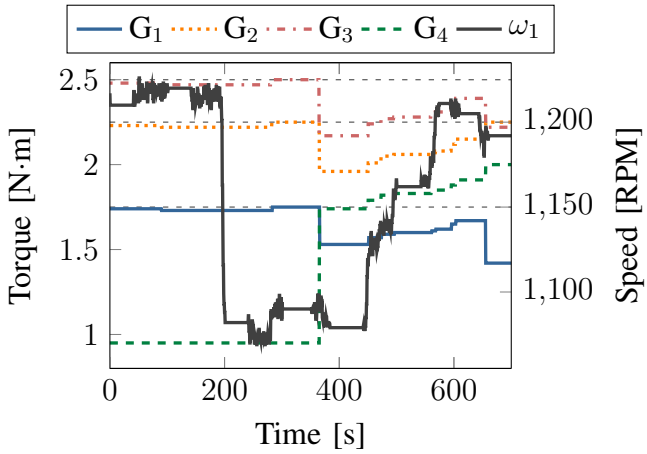


Fig. 14: System response before and after spinning reserve switch-in.

VII. Concluding Remarks

In this paper, we proposed a distributed architecture for generation control in small-footprint power systems, e.g., microgrids. While such power systems are smaller and have lower ratings than their larger counterparts, i.e., bulk power transmission systems, the control objectives are similar; thus, the control functions of our architecture were derived from the three control functions provided by generation control architectures commonly adopted in larger power systems; these functions are (i) droop control, (ii) frequency regulation, and (iii) optimal dispatch.

While droop control is completely decentralized, the implementation of the frequency regulation and optimal dispatch functions is typically centralized. However, by relying on local measurements and simple computations using local information obtained from neighboring generating units, we are able to achieve the same control objectives as those achieved by a centralized implementation. Compared to centralized ones, our distributed control approach can more easily adapt to changes, allowing the system to operate regardless of additions or removals of generating units.

A major component of this research was to experimentally verify the effectiveness of the proposed control architecture to control a small-footprint power system. To this end, we built a microgrid comprised of several small synchronous generators and resistive loads all connected in a ring network. In this microgrid, the exchange of information among generating units was achieved via a wireless communications network, which enabled the implementation of our distributed generation control algorithms for frequency regulation and optimal dispatch. We utilized this microgrid to verify the performance of these algorithms under numerous scenarios, including one where one of the generators, which was acting as spinning reserve, was able to switch in after detecting that the collective generation capacity was not able to match the demand.

APPENDIX A
ELECTRICAL NETWORK HARDWARE DATA

TABLE I: Value of added inductances in power system

Parameter	Inductance [mH]
$L_{1,A}$	2.041
$L_{1,B}$	1.905
$L_{1,C}$	1.961
$L_{2,A}$	4.175
$L_{2,B}$	4.162
$L_{2,C}$	4.059

TABLE II: Hampden Engineering Corporation Synchronous Machine

Parameter	Value
Armature Voltage	133/230 RMS Volts
Armature Current	15.5/9 RMS Amps
Horsepower	2 Hp
Speed	1200 rpm
Frequency	60 Hz
Model	Syn-2

TABLE III: Kollmorgen Goldline Brushless Permanent Magnet Servomotor

Parameter	Value
Stall Current (Continuous)	10.3 RMS Amps
Stall Current (Peak)	33.0 RMS Amps
Torque (Continuous)	6.44 N·m
Torque (Peak)	19.5 N·m
Rated L/L Voltage	230 RMS Volts
Torque (Continuous)	6.44 N·m
Maximum Speed	4900 rpm
Frequency	164 Hz
Model	B-206-C-21

APPENDIX B
RATIO CONSENSUS IMPLEMENTATION DETAILS

Rather than broadcast the latest state values as in (4)–(5), nodes participating in the modified ratio-consensus algorithm broadcast the sum of the weighted states up to and including the current iteration k. For the case when all links are available, i.e., no packets are lost, the weighted states for iteration k can be inferred from the information a DGR receives from its in-neighbors. If a link is temporarily unavailable, however, the modification to the algorithm allows the receiving nodes to recover any lost information at the next successful iteration.

As before, each node maintains two states, $y_i[k]$ and $z_i[k]$, that are updated at each iteration. Using the modified algorithm, however, each node maintains two additional states, $\mu_i[k]$ and $\sigma_i[k]$, which are the values broadcasted to the out-neighbors of DGR i at iteration k. The values of $\mu_i[k]$ and $\sigma_i[k]$ are the sum of $y_i[k]/\mathcal{D}_i^+$ and $z_i[k]/\mathcal{D}_i^+$ since the iterative process began, and thus, they are updated as follows:

$$\mu_i[k+1] = \mu_i[k] + \frac{1}{\mathcal{D}_i^+} y_i[k] = \sum_{t=0}^{k} \frac{1}{\mathcal{D}_i^+} y_i[t], \tag{30}$$

$$\sigma_i[k+1] = \sigma_i[k] + \frac{1}{\mathcal{D}_i^+} z_i[k] = \sum_{t=0}^{k} \frac{1}{\mathcal{D}_i^+} z_i[t], \tag{31}$$

with $\mu_i[0] = 0$ and $\sigma_i[0] = 0$. To account for the fact that the values received from in-neighbors are summations, each DGR i updates its states as

$$\begin{aligned} y_i[k+1] &= \frac{1}{\mathcal{D}_i^+} y_i[k] + \sum_{\substack{j \in \mathcal{N}_i^- \\ i \neq j}} (\nu_{ij}[k+1] - \nu_{ij}[k]), \\ z_i[k+1] &= \frac{1}{\mathcal{D}_i^+} z_i[k] + \sum_{\substack{j \in \mathcal{N}_i^- \\ i \neq j}} (\tau_{ij}[k+1] - \tau_{ij}[k]), \end{aligned} \tag{32}$$

where the values of $\nu_{ij}[k+1]$ and $\tau_{ij}[k+1]$ depend on the successful receipt of a packet from DGR j during iteration k, and are given by

$$\begin{aligned} \nu_{ij}[k+1] &= \begin{cases} \mu_j[k+1], & \text{if } (i,j) \in \mathcal{E}[k], \\ \nu_{ij}[k], & \text{if } (i,j) \notin \mathcal{E}[k], \end{cases} \\ \eta_{ij}[k+1] &= \begin{cases} \sigma_j[k+1], & \text{if } (i,j) \in \mathcal{E}[k], \\ \eta_{ij}[k], & \text{if } (i,j) \notin \mathcal{E}[k]. \end{cases} \end{aligned} \tag{33}$$

[Recall that $(i,j) \in \mathcal{E}[k]$ if DGR i can receive information from DGR j at iteration k.]

Upon examining (33), we see that in order to implement the robust ratio-consensus algorithm, each node must store the most recent successfully received values, necessitating a unique identifier for each node in the in-neighborhood as well as a list of the identifiers of all possible in-neighbors given all available communication links. To identify the senders in our communication and computation testbed, we utilize the 64-bit hardware address associated with the XBee modules which is included in the xbee-arduino API headers. Furthermore, in our setup, the list of possible in-neighbors is programmed on the devices rather than generated at runtime.

Another important implementation detail that is evident from (33) is the fact that the previous successfully received values from each in-neighbor must be saved. In particular, each local controller i must store $\nu_{ij}[k]$ and $\eta_{ij}[k]$ for all $j \in \mathcal{N}_i^-$. While these values can be overwritten upon successfully receiving a packet from an in-neighbor, we see that an additional variable must be utilized for each numerator and denominator of ratio consensus, which, in the case of the distributed optimal dispatch algorithm, amounts to $2n+1$ extra variables.

To account for both of the foregoing characteristics of robust ratio consensus, an object-oriented approach is used for all software written to implement the algorithms. In particular, classes are defined to encapsulate the methods and properties of the graph representing the communication network interconnecting local controllers as well as each of the remote nodes and the local node. Utilizing an object-oriented approach obviates the need to keep track of array indices and allows the software to be congruent with the characteristics of and interconnections between the devices for which it is designed to control.

APPENDIX C
SYNCHRONIZATION PROTOCOL DETAILS

To initiate the time synchronization process, one of the nodes, referred to as the reference node, broadcasts a `sync_begin` packet at time t_1, specifying a target node from its out-neighborhood chosen randomly. The target node then responds using a unicast packet that contains the time the `sync_begin` packet was received, t_2, and the time the response packet was sent, t_3. All other nodes interested in synchronizing to the reference node record the local time at which the `sync_begin` packet was received, t_2', but do not respond. At time t_4, the reference node receives the response packet from the target node and thus owns all of the timestamps required to determine the offset between its local clock and the local clock of the target node. Assuming negligible propagation delay, the reference node computes the offset as

$$d = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \tag{34}$$

and broadcasts it in a final packet also containing t_2.

At this point, the synchronization process for the target node is complete since the synchronized time can be computed as $T = t + d$, where t is the local clock reading. For all other nodes, the timestamp t_2 included in the final packet from the reference node is used to estimate the offset between their local clocks and the local clock of the target node as $d' = t_2 - t_2'$. Using this estimate, the remaining nodes can compute the synchronized time as $T = t + d + d'$, where t is the local clock of the respective node. In the testbed, rather than adjust the clocks of synchronized nodes, a function extending the low-level clock `timer0_millis` is used which adds the offset found using HRTS to the local time, providing a clock that is common throughout the network.

As mentioned above, the computation of the clock offset found using HRTS assumes that the delay due to propagation of messages between devices is negligible. Thus, to minimize synchronization errors, packets should be time-stamped at the lowest protocol layer to reduce delay resulting from data propagating up the stack. In our testbed however, the bottom layer of the stack cannot be modified, so all time stamps are generated at the middle protocol layer. Given this configuration, the delay present in the system results in a worst case clock error on the order of 10 ms. To mitigate the effects of this error on the distributed algorithms, the nodes are restricted from transmitting data for a period of time which exceeds the clock error during the beginning and end of each iteration period.

Following the synchronization of the clocks, one of the nodes must initiate the distributed algorithms to ensure that all nodes in the network begin execution at roughly the same time. To facilitate this, the first node that detects that the frequency has exceeded the specified bound or that the DGRs should be optimally dispatched will broadcast a scheduling packet containing information about the algorithm used, the start time, the number of iterations, and the period of each iteration. Upon receiving the scheduling packet, all other nodes will rebroadcast it to allow the information to propagate throughout the network, then wait until the scheduled start time. If multiple scheduling packets are received, the one with the earliest start time is chosen.

REFERENCES

[1] R. Lasseter, A. Akhil, C. Marnay, J. Stephens, J. Dagle, R. Guttromson, S. A. Meliopoulous, R. Yinger, and J. Eto, "Integration of distributed energy resources: The CERTS microgrid concept," Lawrence Berkeley National Laboratory, Tech. Rep. LBNL-50829, Apr. 2002.

[2] J. Pecas Lopes, C. Moreira, and A. Madureira, "Defining control strategies for microgrids islanded operation," *IEEE Transactions on Power Systems*, vol. 21, no. 2, pp. 916–924, May 2006.

[3] F. Katiraei and M. Iravani, "Power management strategies for a microgrid with multiple distributed generation units," *IEEE Transactions on Power Systems*, vol. 21, no. 4, pp. 1821–1831, Nov. 2006.

[4] A. Tsikalakis and N. Hatziargyriou, "Centralized control for optimizing microgrids operation," *IEEE Transactions on Energy Conversion*, vol. 23, no. 1, pp. 241–248, Mar. 2008.

[5] A. Emadi and M. Ehsani, "Aircraft power systems: technology, state of the art, and future trends," *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 1, pp. 28–32, Jan 2000.

[6] J. Rosero, J. Ortega, E. Aldabas, and L. Romeral, "Moving towards a more electric aircraft," *IEEE Aerospace and Electronic Systems Magazine*, vol. 22, no. 3, pp. 3–9, Mar. 2007.

[7] K. Butler, N. Sarma, and V. Ragendra Prasad, "Network reconfiguration for service restoration in shipboard power distribution systems," *IEEE Transactions on Power Systems*, vol. 16, no. 4, pp. 653 – 661, Nov. 2001.

[8] A. Monti, D. Boroyevich, D. Cartes, R. Dougal, H. Ginn, G. Monnat, S. Pekarek, F. Ponci, E. Santi, S. Sudhoff, N. Schulz, W. Shutt, and F. Wang, "Ship power system control: a technology assessment," in *Electric Ship Technologies Symposium, 2005 IEEE*, July 2005, pp. 292–297.

[9] T. Gruzs and J. Hall, "Ac, dc or hybrid power solutions for today's telecommunications facilities," in *International Telecommunications Energy Conference*, 2000, pp. 361–368.

[10] A. Wood and B. Wollenberg, *Power Generation, Operation, and Control*. New York, NY: Wiley, 1996.

[11] P. Sauer and A. Pai, *Power System Dynamics and Stability*. Upper Saddle River, NJ: Prentice Hall, 1998.

[12] D. B. West, *Introduction to Graph Theory*, 2nd ed. Prentice Hall, 2001.

[13] A. D. Domínguez-García and C. N. Hadjicostis, "Coordination and control of distributed energy resources for provision of ancillary services," in *Proc. IEEE SmartGridComm*, 2010, pp. 537–542.

[14] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed algorithms for control of demand response and distributed energy resources," in *Proc. IEEE Conference on Decision and Control*, 2011, pp. 27–32.

[15] A. S. Debs, *Modern Power Systems Control and Operation*. Boston, MA: Kluwer Academic Publishers, 1988.

[16] A. Bergen and V. Vittal, *Power System Analysis*. Upper Saddle River, NJ: Prentice Hall, 2000.

[17] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY: Cambridge University Press, 2004.

[18] D. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 2004.

[19] M. Madrigal and V. Quintana, "An analytical solution to the economic dispatch problem," *IEEE Power Engineering Review*, vol. 20, no. 9, pp. 52–55, Sep. 2000.

[20] A. D. Domínguez-García, S. T. Cady, and C. N. Hadjicostis, "Decentralized optimal dispatch of distributed energy resources," in *Proc. IEEE Conference on Decision and Control*, 2012, pp. 3688–3693.

[21] Arduino. [Online]. Available: http://www.arduino.cc

[22] Arduino Mega 2560. [Online]. Available: http://arduino.cc/en/Main/ArduinoBoardMega2560

[23] Digi international inc. [Online]. Available: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module.jsp

[24] Sparkfun electronics. [Online]. Available: http://www.sparkfun.com/

[25] Atmel ATmega2560. [Online]. Available: http://www.atmel.com/devices/ATMEGA2560.aspx

[26] ZigBee Alliance. ZigBee Specification. [Online]. Available: http://www.zigbee.org

[27] A. Rapp. xbee-arduino. [Online]. Available: http://code.google.com/p/xbee-arduino/

[28] A. D. Domínguez-García, C. N. Hadjicostis, and N. Vaidya, "Resilient networked control of distributed energy resources," *IEEE Journal on Selected Areas in Comm.*, vol. 30, no. 6, pp. 1137–1148, Jul. 2012.

[29] H. Dai and R. Han, "Tsync: a lightweight bidirectional time synchronization service for wireless sensor networks," *ACM SIGMOBILE Mobile Computing Communications Review*, vol. 8, pp. 125–139, Jan. 2004.