Final Report - February 5, 2009
Advanced Distribution and Control for Hybrid Intelligent Power Systems

M.D. Lemmon - University of Notre Dame

Summary: This report documents the development of a distributed control architecture for advanced distribution and control of mesh microgrids. The resulting architecture is a two-level hierarchy in which the lowest level uses local microsource controllers to maintain microgrid transient stability and the highest level uses a set of *intelligent* agents to optimally manage generator set points and load connectivity. Simulations of the distributed control architecture suggest that it provides a promising method for the "plug-and-play" management of mesh microgrids.

1 Introduction

Microgrids [5] are power generation/distribution systems in which users and generators are in close proximity. This results, in general, in relatively low voltage grids (few hundred kVA). Generation is often done using renewable generation sources such as photovoltaic cells or wind turbines. Power generation is also accomplished through small microturbines and gas/diesel generators. Storage devices such as battery banks represent another important power source for microgrids.

Microgrids are often connected to a main power grid through an intelligent coupling switch. This switch can disconnect from the main grid when the power quality from that grid is no longer acceptable. This results in the *islanding* of the microgrid and a major challenge involves assuring the microgrid's transient stability in the presence of such disconnect events.

Microgrids are often built up in an ad hoc manner. An existing microgrid may be augmented with additional loads and generators as power needs change. It is important that such generation and loads be added in a *plug-and-play* or modular manner. This means that adding the new unit does not require a massive reconfiguration of the existing microgrid controllers. The purpose of this project was to identify a microgrid control architecture that could be augmented in such a modular manner.

This project developed a two-layer hierarchical control architecture for mesh microgrids. The architecture consisted of low-level microsource controllers that use local terminal measurements of voltage and current (power) to assure the entire grids transient stability. This low-level microsource controller is highly modular. It was developed by R. Lasseter for the CERTS (University of Wisconsin, Madison) microgrid. The top level of the proposed microgrid control architecture consists of a set of *supervisory* agents. These "intelligent" agents monitor the global operation of the grid and use that information to adaptively reconfigure both generation assets and load connectivity. In particular, these agents determine the power generation set points for the distributed generation resources and they determine whether or not it is safe for loads to connect to the microgrid. An agent is attached to each load and generation resource. The agents communicate with each other over a wireless mesh radio network in a way that allows them to cooperatively manage the microgrid.

The proposed distributed control concept was implemented on a Matlab/Simulink simulation of a microgrid that was developed by P. Chapman (Univeristy of Illinois, Urbana-Champaign). Simulations were done for an agent-based economic dispatch problem and a load-shedding scenario. Both simulations suggest that the use of intelligent agents provide a feasible way of managing microgrid operations in a plug-and-play manner.

The remainder of this report is organized as follows. Section 1 describes the mesh microgrid that was used throughout this project. Section 2 describes the microsource controllers comprising the lowest level of the system hierarchy. Section 3 discusses issues associated with the use of intelligent agents for the coordinated management of spatially-distributed systems. Section 4 describes the use of such agents to reconnect dropped loads back to the microgrid. Section 5 describes the use of intelligent agents to solve the "economic dispatch problem" in microgrids. Final remarks will be found in section 6. A technical appendix will be found at the end of this document. The appendix provides a more detailed technical overview of the techniques used to implement the economic dispatch problem. The appendix also contains a simulink report of the two simulations that were used to generate the results in this report as well as the listing of the S-functions used in these simulations.

2 Mesh Microgrid

The mesh microgrid being used throughout this project is shown in figure 1. This system consists of three power buses. Bus 1 is connected to the main grid through an intelligent coupler. Bus 1 is also connected to an inductive load $(4 + 0.001s)$ and a 15kW inverter based microsource. Bus 1 is connected to Bus 2 and Bus 3. Bus 2 and Bus 3 are connected to each other. Bus 2 is connected to another 15kW inverter based microsource and Bus 3 is connected to another inductive load $(4 + 0.001s)$. The transmission lines connecting the buses are primarily resistive loads of 0.06045 ohms. All microsources have an interval inductance of 0.00188. They are designed to maintain 120 volts at 60 Hz. The nominal requested real power setpoints are 40 percent of full capacity and the nominal reactive power setpoints are set at 100 percent of full capacity. The main line connecting the microgrid to the primary power grid has a shunt resistance of 10 ohms. The line resistance is 0.052 ohms with an inductance of $0.0231/\omega_0$, where ω_0 is the desired frequency of 60 Hz.

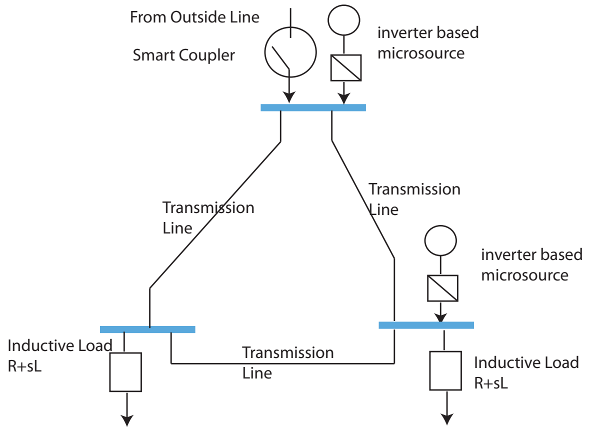


Figure 1: Baseline Mesh Microgrid

A Matlab/Simulink simulation of the system in figure 1 was developed by Dr. P. Chapman at the University of Illinois, Urbana-Champaign (UIUC). The resulting simulink model was based on the CERTS [5] microgrid that was developed by Dr. R. Lasseter at the University of Wisconsin, Madison (UWM). The UIUC simulink model (`genmodel7`) included the microsource inverter controls developed by Dr. Lasseter's group. The load models were modified at the University of Notre Dame (ND) to correctly model the way that loads are shed under frequency droops (`genmodel7_load_shedding`). The Matlab/Simulink Report for this modified simulation will be found in the appendix.

This simulink model simulated the disconnect of the microgrid from the main line. This disconnect occurs at 0.25 seconds and the entire simulation run was 1 second. In this simulation we set the load shedding frequency thresholds at 59.8 Hz. Figure 2 shows the response of the microsources in this scenario. The left hand plots show the responses of generator 1. Arranged from top to bottom are voltage, current, frequency, and power (real and reactive) as a function of simulation time. Similar plots are seen for generator 2. The response of both loads is shown on the right hand side of the figure. From top to bottom are shown the current and real power of load 1 and the current and real power of load 2. The plots show that the frequency stays about 60 Hz, so that no load shedding was observed in this baseline run. The plots show an impulse at 0.25 seconds in the generator voltage and power as a result of the disconnect from the main grid. After the disconnect, the current and power levels at the loads are maintained with very little disturbance, thereby indicating that the microsource control scheme used on the CERTS microgrid is able to provide high quality power through islanding of the microgrid.

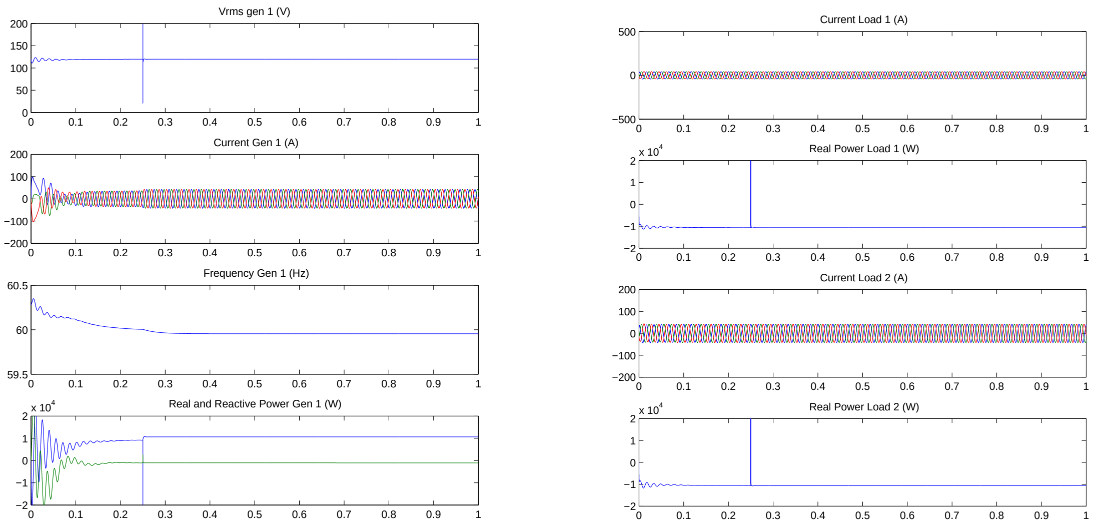


Figure 2: Baseline Scenario Generator Response (right - generator 2 and left - generator 1)

The microsource controllers developed by UWM for the CERTS microgrid were integrated into the microgrid simulation by UIUC. These controllers represent the lowest level of the proposed distributed generation control architecture. A detailed account of the microsource controllers will be found in [4]. For the sake of completeness, the basic operation of this controller is described below.

The inverter-based microsource consists of a D.C. source whose outputs are transformed into an A.C. voltage through an inverter. The actions of the inverter are guided by a controller that uses sensed feeder currents and voltages to determine how best to control the operation of the inverter. Figure 3 shows that the output of the inverter is passed through a low pass filter to remove switching transients to produce a three phase 480V voltage. A transformer then steps this down to 208 V (120 volts rms).

The UWM microsource controller is shown on the right hand side of figure 3. The inputs are measurements of inverter current, load voltage and line current. The controller also takes as reference

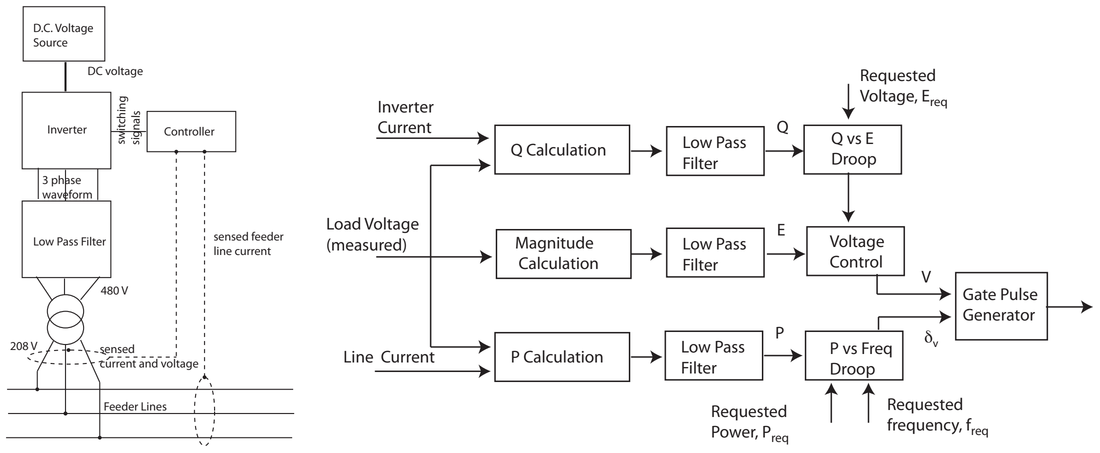


Figure 3: (Left) - inverter-based microsource - (Right) - UWM microsource controller

inputs the requested voltage level $E_{\rm req}$, the requested power set point $P_{\rm req}$, and the desired frequency $f_{\rm req}$ (usually 60 Hz). The controller takes the measured inputs and computes the instantaneous reactive power, Q, the voltage magnitude, E, and the real power P. These computed values are low pass filtered. The reactive power, Q, and the requested voltage $E_{\rm req}$ are input to the Q vs E droop controller to determine the desired voltage level. This is compared against the measured voltage level and the output V is then given to the gate pulse generator. Another channel in the controller uses the measured real power and implements another droop control that balances the system's frequency against the requested power level, $P_{\rm req}$. The output of the P vs frequency droop controller is used to adjust the phase offset, δ_V, which is also fed into the gate pulse generator. The output of the gate pulse generator goes directly into the inverter.

The action of this controller is, essentially, to mimic the droop controls seen in traditional synchronous generators. This means that if a load begins drawing a great deal of real power, then the line frequency will "droop" as an indicator of the extra stress on the system. The controller automatically tries to restore that frequency to its desired levels. But it will be unable to restore the droop if the power being pulled if the power drawn by the load exceeds the generator's capacity. This drop in frequency can be sensed at the load and may be used to help decide if the load should disconnect from the microgrid. A similar scenario occurs if the load begins drawing too much reactive power. In this case, there will be a droop in the voltage that can again be used by the load to determine if it should disconnect from the grid.

3 Distributed Supervisory Control though Intelligent Agents

The mesh microgrid with its hierarchical control system constitutes what is sometimes referred to as a *multi-agent networked system.* Such systems consist of two types of subsystems; a collection of *physical processes* and another collection of computational or *cyber* processes. The physical processes usually interact through direct physical interactions, so that these physical processes form a network. For a microgrid, this network of physical processes consists of the microsources and loads that are interconnected through the transmission lines. The cyber processes are sometimes referred to as *intelligent agents.* These agents communicate over a digital communication network. In our case, the resulting network of cyber processes consists of the microcontrollers that are used to decide generator set points.

Figure 4 illustrates the two networks comprising the multi-agent system. The network of physical processes are shown by rectangles that are connected by the solid lines. Each rectangle represents a physical process. The line represents the physical interaction between the processes. Each process has a local state that is represented by a continuous-time function, $x_i(t)$. This local state evolves according to the differential equation

$$\dot{x}_i(t) = f_i(x_i(t), x_{-i}(t)) + g_i(u_i(t)) \tag{1}$$

From equation 1, it is apparent that the dynamics of the ith physical process are governed by two functions, f_i, and g_i. The first function, $f_i(x_i, x_{-i})$ models the physical interconnection between the ith process and its neighbors. The first argument of f_i represents the ith agent's local state and the second argument represents the impact that neighboring agent states have on the ith physical process. The interconnections embodied in f_i are of a continuous nature.

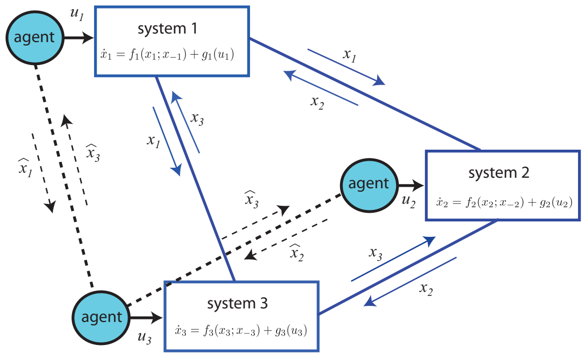


Figure 4: Multi-agent System

The other term driving the process state is given by the function, g_i. This function represents the impact that the cyber-process (agent) has on the ith agent's local state. As shown in figure 4, an agent is associated with each physical process. The agent generates a continuous-time control signal $u_i(t)$ that is the argument driving the second term in equation 1. This "control" signal is a function of the state information that is received from neighboring agents over a digital communication network. The links of this communication network are shown as dashed lines connecting the agents in figure 4.

The agents broadcast the physical process state over a digital communication network. Because the links are digital, messages can only be broadcast at discrete time instants. So unlike the "continuous-time" connections driving the physical process states, the state information transmitted between agents is a discrete-time signal. In particular, we can model these transmitted signals as a sequence of *events*. Each event is an ordered pair $(b_i[k], \hat{x}_i[k])$ for $k = 0, \ldots, \infty$. $b_i[k]$ denotes the time when agent i broadcasts for the kth consecutive time and $\hat{x}_i[k]$ denotes the state information that was broadcast at that time. This transmitted state information is the physical process' state at time $b_i[k]$, so that $\hat{x}_i[k] = x_i(b_i[k])$. The ith agent uses the sampled states of its neighbors, $\hat{x}_{-i}$ to compute the control input $u_i(t)$. This is done using a controller function k_i so that

$$u_i(t) = k_i(\hat{x}_{-i}[k]) \tag{2}$$

for $t \in [b_i[k], b_i[k+1])$ for all k and where $\hat{x}_{-i}$ denotes the broadcast states of the ith agent's neighbors.

Since information is transmitted in a discrete manner between agents, the frequency with which such transmissions occur can greatly impact how well these agents coordinate their actions. In particular, if agents communicate very infrequently, they will not have enough information to adequately coordinate their actions. On the other hand, if communication is very frequent, then the cost associated with maintaining such a high transmission rate may be unrealistic for a real-life system. This is particularly true in wireless communication networks where fading and multi-user interference can degrade a link's throughput in an unpredictable manner. The unreliability of network communication, therefore, represents a major issue that must be addressed in the design of multi-agent systems.

One way of addressing this issue is to find ways that balance the cost associated with data transmission against the actual performance of the overall system. In particular, one can maximize the period between successive broadcasts in a way that preserves the overall system's performance. This objective can be realized by conditioning the broadcast of agent information on the "importance" that information has on the system's performance. This approach is sometimes referred to as *event-triggering* [1] [6].

Event-triggering initiates broadcasting when some local "event" occurs. The basic principle is that agents should only broadcast when there is innovative or novel information in the local state. In other words, the agent only broadcasts when the process is doing something unexpected. Recent work [9] has shown that simple threshold conditions such as

$$\|x_i(t) - \hat{x}_i[k]\| \gtrless \rho \|x_i(t)\| \tag{3}$$

can be used to initiate message broadcasts in a way that preserves overall system performance while greatly reducing the average broadcast period. The above condition triggers a broadcast when the error between the process' local state $x_i(t)$ and the last broadcast state $\hat{x}_i[k]$ exceeds a threshold. This threshold, itself, is a function of the process' current state and a heuristic justification of this state-dependent threshold will be found in the appendix.

The supervisory control architecture used by this project to manage mesh microgrids is a multi-agent system similar to what's shown in figure 4. The agents use event triggering (equation 3) to reduce the number of broadcasts between agents. The following sections detail the event-triggered multi-agent achitectures that were developed under this project to 1) determine generator set points and 2) determine whether or not to reconnect a previously shed load.

4 Intelligent Load Shedding

The mesh microgrid from figure 1 was augmented with intelligent agents monitoring the quality of the power being produced by the generators. These real-time measures of power quality were then used to trigger the safe reconnection of previously shed loads to the microgrid. Figure 5 shows our original mesh microgrid with additional agents that are used to control load reconnection. There are two types of agents; *power quality* agents and *load* agents. A power quality agent is attached to each generator and keeps track of the "quality" of the power being delivered by that generator. The *load* agent is attached to each load and is used to enable the reconnection of that load to the grid.

The power quality agent monitors the quality of the microsource's power to determine whether or not it is same for loads to connect to the generator. There are many ways of monitoring power quality.

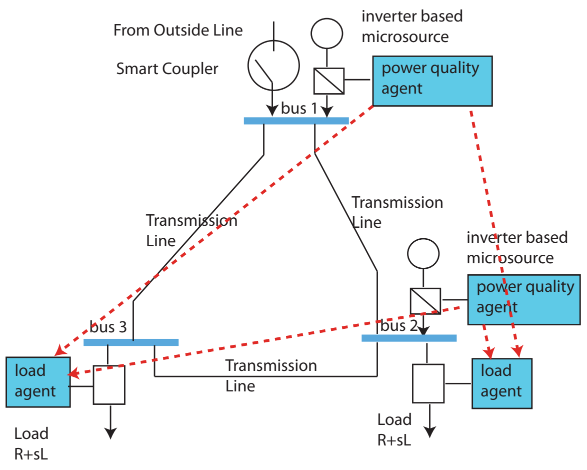


Figure 5: Baseline Mesh Microgrid with Load Controls

In this scenario we developed simple nonlinear filters whose output grows when the generator power exhibits a significant oscillation. The power quality filter consists of a differentiator with a full-wave rectifier. We then pass this signal through a low pass filter. The result is a signal whose value is close to zero when the measured power is not oscillating (high power quality) and whose value is large if there are significant oscillations (i.e., poor power quality) in the generated power. The output of this low pass filter is then passed through a hysteresis function that generates a boolean TRUE signal when the power quality is high and a boolean FALSE signal when the power quality is poor. Essentially, we can view the power quality agent as a boolean system that detects whether the generator power quality is good or bad.

A power quality agent is associated with each generator, thereby producing two distinct logical assessments of the network's power quality. These two boolean signals are then transmitted to the load agent. Note that the agent only needs to broadcast the signal when its changes, so the broadcasts from the power quality agents are event-triggered. The load agent simply takes the AND of both signals. If the output of the AND operation is TRUE, then it is safe for the load to reconnect to the grid and the load agent issues a RESET signal to the load. If it is false then it is unsafe for the load to reconnect to the grid and no RESET signal is issued by the load agent. The resulting network of agents forms a distributed event-triggered detection system.

The system shown in figure 5 was implemented in the UIUC mesh microgrid simulation. The changes involved adding the power quality agents and the load agents to the simulation. It was also necessary to make small changes to the load models. In particular, the load model was changed so that load 1 would exhibit a ground fault at 0.5 seconds. The load models were also changed so that the load would automatically disconnect from the grid if the frequency fell below 59.8 Hz. Once disconnected, the load would not reconnect to the grid unless it recieved a RESET signal from the load agent, indicating that it was safe to reconnect.

The simulation was used to test a specific load shedding scenario. Under the scenario, the microgrid is initially connected to the main grid until 0.25 seconds, after which a disconnect occurs from the main grid. The next event occurs at 0.5 seconds into the simulation, when load 1 exhibits a ground fault. The ground fault causes the frequency of both loads to dip below 59.8 Hz, thereby causing both loads to be shed from the microgrid. The final significant event occurs at 0.7 seconds into the simulation, when the load agents signals that it is alright for the second load to reconnect to the grid.

This sequence of events can be observed in the plots of figure 6. Figure 6 shows simulation results for generator 1 and the two loads. From top to bottom, the plots on the left hand side show generator 1's rms voltage, current, frequency and power (real and reactive). From top to bottom the plots on the right hand side show the first load's current and real power and the second load's current and real power. The main disconnect at 0.25 seconds is marked by the impulse in generator 1's rms voltage at 0.25 seconds. Similar impulses are seen in the generator and load power plots. The ground fault on load 2 is observed in the top left plot of the figure. This plot shows the current drawn by load 1 and it exhibits a significant step change at 0.5 seconds. As mentioned above, this fault causes a drop in the frequency as shown in generator 1's frequency plot on the left hand side of figure 6. This droop in frequency also causes load two to drop from the network which can be seen in the current plot for load 2 on the right hand side of the figure. In this figure we see the current drawn by load 2 drops to zero between 0.5 and 0.8 seconds. At 0.7 seconds, the second load reconnects to the grid.

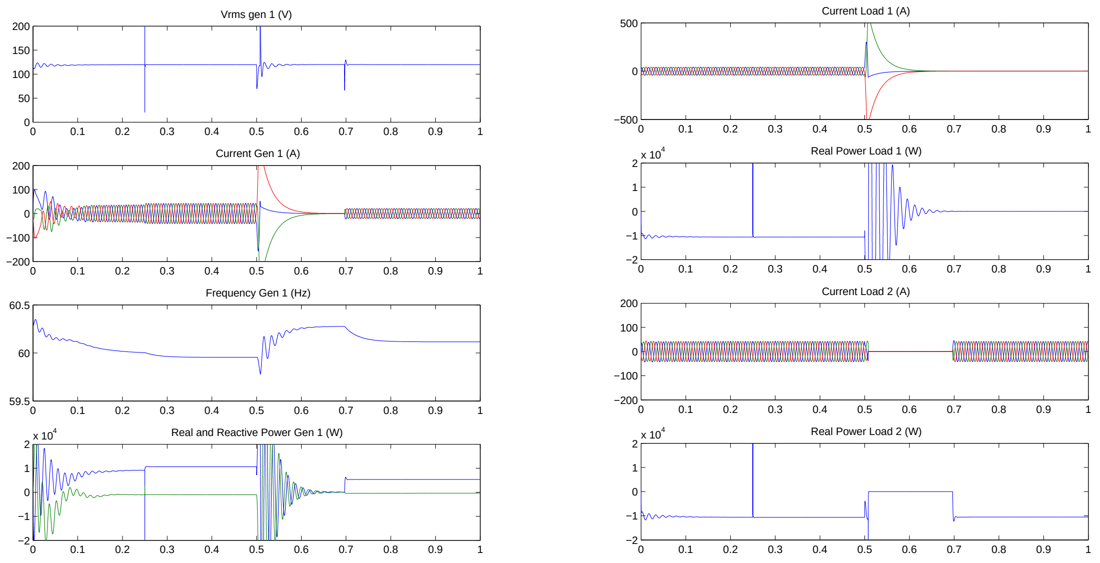


Figure 6: Load Shedding Scenario - Generator Time Histories - Right hand plots are for loads 1 and 2. Left hand plots are for generator 1.

It is apparent that the ground fault at 0.5 seconds results in significant oscillations in generator power (right hand side of figure 6). So for a certain interval of time after the ground fault, the quality of the power produced by both generators is relatively poor. The power quality agent uses a simple nonlinear filter to measure the power quality. The block diagram for this filter is shown on the right side of figure 7. The time histories of the generator 1 and 2's power quality are shown in figure 7. As expected, power quality is poor when the microgrid is first switched and when the ground fault occurs. The bottom plot in figure 7 shows load agent 2's RESET signal as a function of time and as expected, load 2 is allowed to reconnect to the microgrid at about 0.7 seconds.

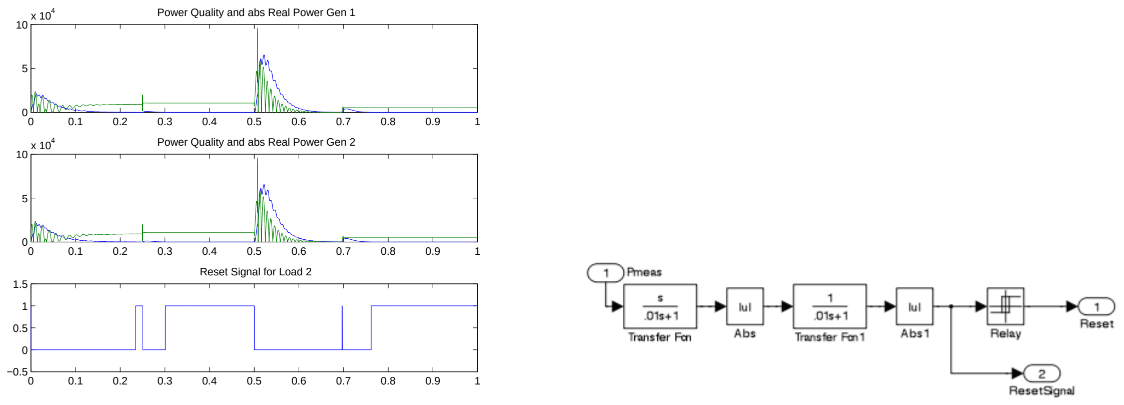


Figure 7: Load Shedding Scenario - Power Quality Agent and Load Agent Response. Right hand block diagram is for power quality agent's filter. Left hand plots from top to bottom are the power quality filter output (ResetSignal) for generators 1 and 2 and the reset signal generated at the load agent.

The simulation results shown in figures 6-7 both indicate that it is feasible to use the proposed multi-agent approach to help manage load shedding in mesh microgrids. The simulations demonstrated that one can use intelligent agents to automate load reconnection with a distributed network of power quality and load agents that communicate in an event-triggered manner. While this study confined its attention to making intelligent reconnect decisions, it is also possible to augment load agent logic to account for mission specific priorities on various loads.

5 Intelligent Power Dispatch

The mesh microgrid from figure 1 was augmented with intelligent agents that adaptively readjust the real power being requested of each generator. These agents coordinate their requests in order to minimize the overall cost of operating the generators. These agents are therefore solving the so-called economic dispatch problem [2] in a distributed manner. Figure 8 shows agents that were added to the microgrid. There are three types of agents; *power agents*, *price agents*, and *load agents*. A power agent is attached to each microsource. The power agent determines what real power levels to be requested from the generator. A *price* agent is associated to a group of generators whose power levels must be constrained to stay within operational limits. The price agent computes a *shadow-price* which help power agents coordinate the selection of their individual power requests. The *load* agent is an agent that monitors the quality of the power at the load and broadcasts forecasted load power levels to the price agent.

These agents work in a coordinated manner to solve the economic dispatch problem. Neglecting line

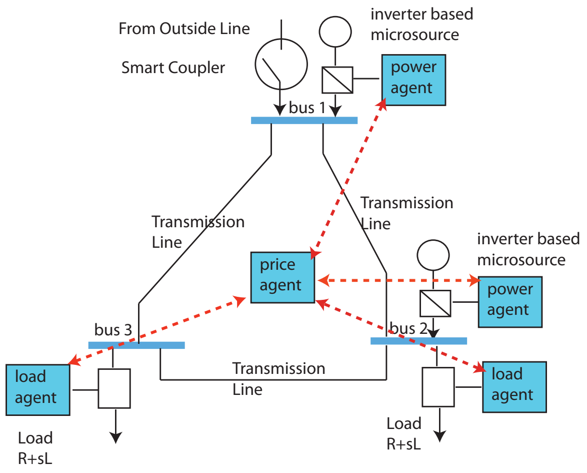


Figure 8: Baseline Mesh Microgrid with Price Control on Real Power

losses, the dispatch problem for the two generators in the microgrid can be stated as

$$\begin{array}{ll} \text{minimize:} & C(P_1, P_2) = \sum_{i=1}^{2} C_i(P_i) \\ \text{with respect to:} & P_1, P_2 \\ \text{subject to:} & \sum_{i=1}^{2} P_i = \sum_{i=1}^{2} P_{Li} \\ & 0 \leq P_i \leq 1, \quad (i = 1, 2) \end{array} \tag{4}$$

where P_i is the power generated by the ith microsource and P_{Li} is the power measured at the ith load. In this case, $C_i : \Re \to \Re$ is the cost associated with operating the microsource at a given power level. In the simulation, these power levels are given in "power units" (pu) that are normalized by the maximum capacity of the microsource. In this simulation, that normalizing constant is 15 kW. The simulations described below use a specific set of cost functions. The costs functions used were

$$C_1(P_1) = 20 + 0.1P_1 + 0.1P_1^2 \tag{5}$$

$$C_2(P_2) = 10 + 0.5P_2 + 0.2P_2^2 \tag{6}$$

The first constraint shown in equation 4 requires that the total generated power equal the total load power. The second constraint requires that the generated power be bounded between 0 and 1 pu.

A multi-agent approach was used to solve the dispatch problem in a distributed manner. The approach takes advantage of the fact that the dispatch problem is very similar to the network utility maximization (NUM) problem [3]. A detailed description of the NUM problem will be found in the appendix. To see how the NUM methodology can be applied to the dispatch problem, it will be convenient to rewrite equation 4 as a NUM problem. Let $p = \begin{bmatrix} P_1 & P_2 \end{bmatrix}^T$ denote a vector of the requested power levels for both microsources. The dispatch problem may then be expressed as

$$\begin{array}{ll} \text{minimize:} & C(p) = \sum_{i=1}^{2} C_i(P_i) \\ \text{subject to:} & Ap \leq c \end{array} \tag{7}$$

where we let

$$A = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \overline{P} \end{bmatrix} \tag{8}$$

and $\overline{P} = \sum_{i=1}^{2} P_{Li}$ represents the maximum load power. Note that this has the same form as the NUM problem in the appendix.

From equation 7, one can construct the associated dual problem that seeks to minimize

$$C_d(p) = \max_{p>0} \left[\sum_{i=1}^{2} C_i(P_i) - \lambda^T (Ap - c) \right] \tag{9}$$

As in the NUM problem, the original cost function $C(p)$ has been augmented with the term $\lambda^T(Ap-c)$ to obtain equation 9. The function $C_d(p)$ may therefore be interpreted as the cost of running the microsouces discounted by how close the power levels p are to violating the inequality constraint $Ap \leq c$. The vector λ can be thought of as a vector of "shadow-prices" that a *price agent* charges each microsource for violating one of the power constraints in equation 4.

Solving the original dispatch problem can then be done by using a gradient-following algorithm to search for the power levels, P_1 and P_2, and prices, λ, such that the discounted cost in equation 9 is minimized. One such algorithm is the *dual-decomposition* algorithm [7], which takes the form

$$\begin{aligned} P_i[k+1] &= arg \max_{P_i} \left[C_i(P_i[k]) - P_i[k] \sum_j \lambda_j[k] \right] & (10) \\ \lambda_j[k+1] &= \max\left(0, \lambda_j[k] + \gamma\left(P_1[k] + P_2[k] - c_j\right)\right) & (11) \end{aligned}$$

for $i = 1, 2$ and $j = 1, \ldots, 5$. In the above equation γ is a step size that must be chosen small enough to assure the convergence of the recursion. Note that the first equation (eq. 10) can be solved in closed form for the quadratic cost functions that were given in equation 5-6. In particular, for these specific cost functions it can be shown that

$$p[k+1] = \begin{bmatrix} P_1[k+1] \\ P_2[k+1] \end{bmatrix} = p[k] - \gamma \left(A^T \lambda[k] + \begin{bmatrix} .1 \\ .5 \end{bmatrix} \right) \begin{bmatrix} 5 & 0 \\ 0 & 5/2 \end{bmatrix} \tag{12}$$

where γ is a step size that must be chosen to assure the convergence of the recursion. For these simulations γ was chosen to be 0.01. So the solution to the dispatch problem by applying the recursion given by equations 11 and 12.

The distributed recursion given by equations 11 and 12 are realized through agents. A *price agent* is used to compute the price in equation 11. From equation 11, it should be apparent that the price agent uses the current power request levels, $P_1[k]$ and $P_2[k]$, and the load levels, P_{L1} and P_{L2}, to compute the shadow-price. It obtains P_1 and P_2 from the power agent and it obtains P_{L1} and P_{L2} from the load agents. The ith power agent computes the P_i using equation 12. Due to the closed form nature of this equation, the power agent only needs the current shadow price vector, λ, which it gets from the price agent. The load agent uses an algorithm to forecast the predicted load power levels, P_{Li}. For this simulation, this forecast was obtained by simply passing the load's measured power levels through a low pass (averaging) filter. Other, more sophisticated, forecasting tools could be used by the load agent to include the effect of diurnal variations on load power.

This set of agents, therefore, are related in a manner that mirrors the functional dependencies in equations 12 and 11. These relationships, of course, are graphically illustrated in figure 8, where the dashed lines show that the load and power agents are all connected through the single price agent in a star topology. The price agent, therefore, represents a single agent that is really responsible for coordinating the decisions made by all of the power agents. This example, therefore seems to suggest that the proposed multi-agent solution actually makes decisions in a centralized manner. This conclusion, however, is not quite true. This example is a small microgrid consisting of two generators and two loads that are tightly interconnected. One can easily see a larger microgrid being composed of several interconnected service areas and for this larger microgrid, the proposed control architecture would have a price agent associated with each service area. Therefore the multi-agent solution being proposed here is indeed a distributed control architecture in which price agents help coordinate the power agents within a single service area of a larger grid.

As noted in section 3, the proposed multi-agent control architecture relies on the transmission of information over digital communication links. Since these links are digital, information is transmitted at discrete time instants and it is important that one maximizes the period between successive broadcasts. Maximizing broadcast period can help reduce the cost of the associated communication infrastructure and it can result in lower message passing latency since there is less competition for the shared communication channels. Long broadcast periods, however, can also adversely effect the ability of agents to coordinate their actions. One must therefore bound the need to reduce the amount of network traffic against the overall performance of the physical system. This project used the event-triggering approach outlined in section 3 and detailed in the appendix to select the time instants when the agents would broadcast their states to each other.

Event-triggering has an agent broadcast its local state when a condition similar to equation 3 is about to be violated. For the price agent, the event-triggering condition is

$$\left\| \lambda(t) - \hat{\lambda}[k] \right\| \leq \rho \| \lambda(t) \| \tag{13}$$

for $t \in [b[k], b[k+1])$. $b[k]$ denotes the kth consecutive time when the price agent broadcast its price to the power agents. $\lambda(t)$ represents the price computed at time t using equation 11 and $\hat{\lambda}[k] = \lambda(b[k])$ represents the price vector that was last broadcast by the price agent. For this simulation ρ was chosen to be 0.01. So the price agent broadcasts it current price, $\lambda(t)$, when equation 13 is about to be violated.

For the power agent, the event-triggering condition is

$$|P_i(t) - \hat{P}_i[k]| \leq \rho |P_i(t)| \tag{14}$$

for $t \in [b_i[k], b_i[k+1])$ where $b_i[k]$ denotes the k consecutive time when power agent i broadcast its power request level, P_i, to the price agent. In the above equation $P_i(t)$ is the power level computed at t using equation 12. Note that this will change each time a new price vector, λ, is received from the price agent, so that $P_i(t)$ is varying in a step-wise manner between consecutive broadcasts from the power agent. $\hat{P}_i[k]$ is the power level that was last broadcast by the ith power agent. In this simulation the event-triggering gain, ρ was chosen to be 0.001. The power agent broadcasts its current requested power level, $P_i(t)$, when equation 14 is about to be violated.

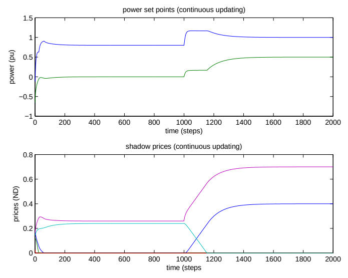
Finally, the event-triggering condition for the load agent was chosen to be a simple constant threshold $|P_{Li}(t) - \hat{P}_{Li}| < \rho$ where $\hat{P}_{Li}$ is the last broadcast load power, P_{Li} is the measured load power and for this simulation the threshold was chosen to be $\rho = 0.01$. In future implementations, P_{Li} should represent a forecast of the future power required by the load. In these simulations, that forecast was obtained by low pass filtering the measured power load. Such low pass filters can be easily modified for use as load forecasters. So the load agent broadcasts its forecasted power level, $P_{Li}(t)$, when that forecast changes by a fixed amount from the previously broadcast power forecast level.

The power dispatch algorithm described above was simulated using a Matlab script under two scenarios. The first scenario assumed that power and price agents had continuous-time access to each other's states. The second scenario assumed that these agents broadcast their states using the event triggers described in equations 13-14. In both scenarios, the total load power requested $\overline{P}$ starts at 0.8 pu and then makes a step change to 1.5 pu 1000 time steps into the simulation. The optimal power allocations when $\overline{P} = .8$ were computed to be $P_1 = .8$ and $P_2 = 0$. The optimal power allocations when $\overline{P} = 1.5$ were computed to be $P_1 = 1$ and $P_2 = .5$. Under the continuous-time access scenario, the power setpoints and prices generated by the algorithm are shown on the left side of figure 9. These plots show that for the chosen set of parameters the algorithm converges as expected to the optimal values mentioned above.

The simulation results for the second scenario, along with the associated Matlab script, will be found in figure 9. The top two plots show the power set points and prices generated by the event-triggered algorithm. Comparing these plots to the results for the continuous-access scenario, it is seen that event-triggering does not significantly degrade the overall algorithm's performance since the plots are nearly identical. The bottom two plots in figure 9 show how many agents are broadcasting at a given time in the simulation. In this case, a single price agent was associated with each inequality constraint. Since there are five inequality constraints, there are actually five price agents in this simulation. What is observed in these last two plots is that all agents are broadcasting when there is a step change in the load power forecast. So there are a lot of agents broadcasting at 0 and 10 seconds into the simulation. After the prices and power set points have settled to their optimal values, the number and frequency of broadcasts reduces dramatically, thereby indicating that the event-triggered dispatch algorithm reduces communication effort while still achieving the optimal power set point allocations.

The preliminary simulation results demonstrated that the event-triggered power dispatch algorithm would work well, but that simulation did not include the dynamics of the low-level microsource controllers developed by UWM. The next set of simulation tasks involved integrating the proposed power dispatch algorithm into the UIUC microgrid simulation. The resulting Matlab/simulink model (`genmodel8_price_control`) is described in the simulink generated report in the appendix. The new blocks that were added to the original UIUC model are highlighted in blue. As can be seen in the appendix, the simulink model was augmented with new subsystems representing the power agents, price agent, and load agents. The load subsystems in the original UIUC model were also modified to properly model load disconnects and to provide measurement of load power to the load agent.

The resulting simulink model was used on a scenario in which the microgrid initially starts connected to the main grid, disconnects from the main grid at 0.25 seconds and then has load one make a step change in its load resistance from 4 to 16 ohms at 0.5 seconds into the simulation. The objective of the simulation was to see whether the UWM microsource controller would interfere with the proposed multi-agent power dispatch algorithm and to evaluate the actual broadcast periods that could be expected in the system.



```
%Initiailization
x0 = rand(2,1);
A=[1 0;-1 0; 0 1;  0 -1;-1 -1];
B2 = [1;0;1;0;-1.5];
B1 = [1;0;1;0;-.8];

%initialize dual-decomp algorithm
p0 = .25.*rand(5,1);p=p0;
x=x0;xhat = x0; phat = p0; p; = p0
gam=.01;B=B1;tstop=2000;tevent = 1000;

%simulate with event-triggered updating
for i=1:1:tstop;
    if(i==tevent);
        B=B2;
    end;

  x = -(A'*phat+[.1;.5])./(2*[.1;.2]);
  zx = (abs(x-xhat)<=.001.*abs(x));
  xhat = zx.*xhat + (1-zx).*x;

  p = max(0,p+gam.*(A*xhat-B));
  sx = (abs(p-phat)<=.01.*abs(p));

  phat = sx.*phat + (1-sx).*p;
  upx = sum(1-sx); vpx = sum(1-zx);
end;
```

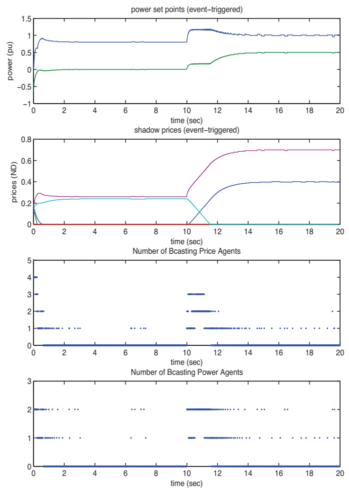


Figure 9: Matlab Script implementing event-triggered power dispatch and simulation results

The following figure shows the time histories of the generator's variables. Generator 2 exhibits similar behaviors. From top to bottom the plots on the left hand side show the rms voltage, current, frequency, and active/reactive powers for generator 1. The main disconnect event at 0.25 seconds is clearly observed by the impulses that appear at 0.25 seconds. In this case, the load shedding thresholds were set to zero to prevent load shedding when the frequency begin dropping after the disconnect. The second event at 0.5 seconds is also marked by impulses at these times. The variations at 0.5 seconds are caused by the power dispatch algorithm readjusting generator setpoints. The plots show that the microgrid operated well during the main switch and during adjustments of set point levels due to the proposed multi-agent control system.

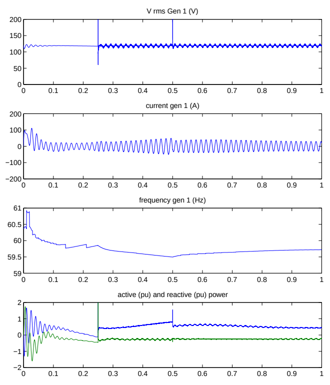


Figure 10: Generator time histories using event-triggered power dispatch

The behavior of the power, price, and load agents will be seen in the plots below. This figure consists of three groups of plots for each set of agents. The plots show the output of the agent as well as the time between broadcasts. All of these plots show that the time between agent broadcasts becomes longer as the system settles to its equilibrium point. So, as expected the event-triggered power dispatch algorithm appears able to maintain satisfactory system operation while reducing the system's usage of the communication network.

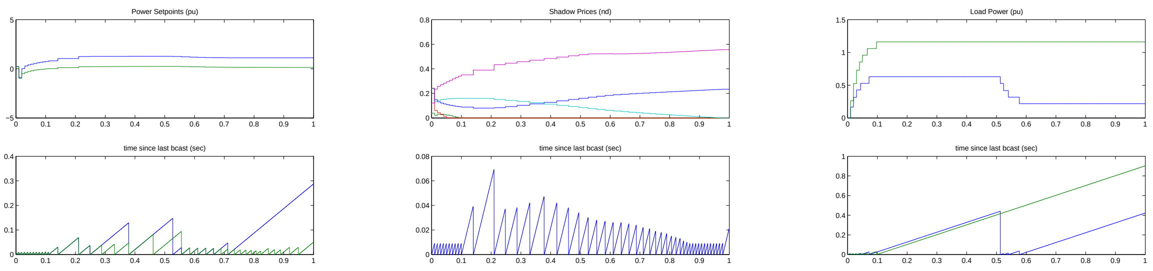


Figure 11: Agent Performance

The results discussed above show that a multi-agent solution to the economic dispatch problem can be built in a way that integrates easily with the UMW microsource controllers used in mesh microgrids. These results further indicate that the broadcast period of such agents can be effectively increased using event-triggered methodologies, without compromising the performance of the overall microgrid. The simulation results presented here are preliminary for they only apply to a small microgrid. It would be useful to determine if the approach scales up well to larger microgrids.

6 Final Remarks

This report documents efforts at the University of Notre Dame to develop a multi-agent control system architecture for a mesh microgrid. The project took an existing simulink model of a mesh microgrid that was developed at UIUC. The microgrid model used low level microsource controllers developed at UWM. The objective was to develop a distributed event-triggered agent based approach to solve the economic dispatch problem and reconnect decision problem. The economic dispatch problem tries to automatically determine generator power setpoints that meet a set of inequality constraints of overall power usage in the microgrid. The reconnect problem seeks to find a distributed decision making algorithm for informing loads that generated power quality is stable enough to permit safe reconnection to the microgrid. The agent-based algorithms developed in this project were simulated on the UIUC simulink model and preliminary results indicate that it is feasible to integrate these agent-based control architectures into mesh microgrids such as the CERTS grid at the University of Wisconsin Madison.

There are numerous directions for extending the accomplishments of this project. Some of these directions are itemized below

- In the first place, the feasibility studies in the project were done on a very small microgrid. It would be valuable to investigate how well the approach scales on larger microgrids.

- The simulations only modeled the discrete-nature of agent broadcasts, it did not model delays or message drops due to unreliable links. It would also be valuable to study the impact such real-life communication issues have on overall system performance.
- The power dispatch problem addressed here only considered real power, not reactive power. This was because the droop controls in the UIUC microgrid simulink model only implemented the real part of UWM's controller. It would be useful to extend the multi-agent approach to dispatch of reactive power as well.
- The reconnect scenario considered in this project was very simple. There are numerous ways such a scenario could be extended to consider the loss and restart of generating assets and how this might be integrated into load reconnection.
- While simulations represent an important first step in assessing the feasibility of any approach, it would be better to test the multi-agent architecture on a real small scale microgrid.

Appendix A: Economic Dispatch Problem [2]

We consider a mesh connected power system that consists of generators and loads connected to a buses that are interconnected through transmission lines. The network of buses and transmission lines is modeled as a connected graph $G = (V, E)$ where $V = \{v_1, \cdots, v_N\}$ is a set of N buses and $E \subset V \times V$ is a set of transmission lines between buses. Figure 12 shows the graph associated with a power distribution network consisting of five interconnected buses. We suppose there are M edges in the network and we let e_{ij} denote the edge from bus i to bus j. We let $Z_{ij} = R_{ij} + jX_{ij}$ denote the impedance of the transmission line corresponding to transmission line e_{ij}. Let I denote the incidence matrix for graph G and define a diagonal matrix $D \in \Re^{M\times M}$ whose diagonal entries are the reactances of the M transmission lines. Then we define the *weighted incidence* matrix $A \in \Re^{M\times M}$ as $A = DI$. The set of neighbors of bus v_i is denoted as $\mathcal{N}_i = \{v_j \in V \mid (v_i, v_j) \in E\}$.

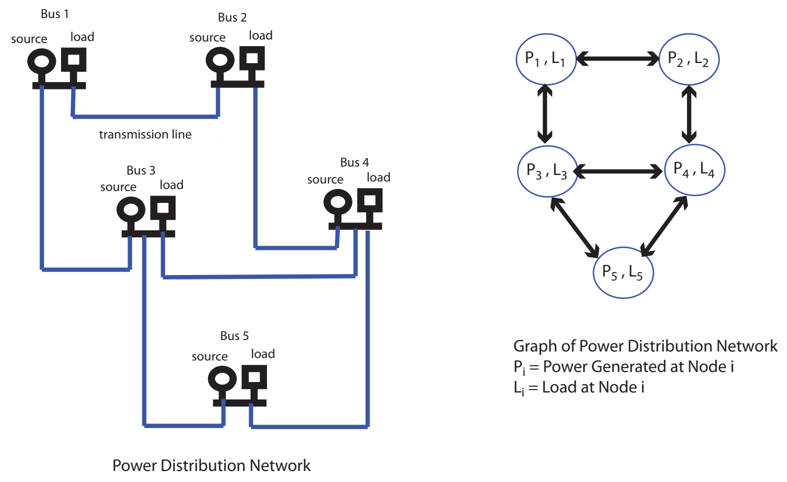


Figure 12: Power Distribution Network and its Graph

Let u_i denote the voltage of bus i. This is a complex number $u_i = |u_i|e^{j\theta_i}$ which represents the amplitude and phase of the signal's primary phase. The complex power flow from bus i to bus j is denoted as S_{ij} and is equal to

$$S_{ij} = u_i \left(\frac{u_i - u_j}{Z_{ij}}\right)^* = P_{ij} + jQ_{ij} \tag{15}$$

where P_{ij} is the real power flow from node i to j and Q_{ij} is the reactive power flow from node i to j. We can solve directly for the real power and reactive power to see that

$$P_{ij} = -P_{ji} = \frac{|u_i||u_j|}{X_{ij}} \sin(\theta_i - \theta_j) \tag{16}$$

$$Q_{ij} = \frac{|u_i|^2}{X_{ij}} - \frac{|u_i||u_j|}{X_{ij}} \cos(\theta_i - \theta_j) \tag{17}$$

$$Q_{ji} = \frac{|u_j|^2}{X_{ji}} - \frac{|u_i||u_j|}{X_{ji}} \cos(\theta_i - \theta_j) \tag{18}$$

Under normal operating conditions, we assume that $|u_i| \approx |u_j|$ and that $\theta_i - \theta_j$ is small. In this case there is reasonably good decoupling between the control of the active power P_{ji} and the reactive power flow Q_{ij}. The active power flow is mainly dependent on $\theta_i - \theta_j$ and the reactive power flow is primarily dependent on $|u_i| - |u_j|$.

The DC power flow assumes that only the voltage phases, θ_i, vary and that this variation is small. The Voltage magnitudes $|u_i|$ are constant and nearly equal so that the reactive power flow Q_{ij} is negligible. As a result, the problem we'll focus on initially only concerns the economic dispatch of the active power flow, P_{ij}. Furthermore, we assume that the transmission line's resistance is negligible so that $Z_{ij} = jX_{ij}$. With these assumptions, the power flow from bus i to bus j is

$$P_{ij} = \frac{1}{X_{ij}}(\theta_i - \theta_j) \tag{19}$$

The total power flowing into bus i (which we denote as P_i) is the power generated at node i (denoted as P_{Gi}) minus the local load (denoted as P_{Li}) at the bus. Based on the conservation of power, this must equal the sum of the power flowing away from bus i on the transmission lines, so that

$$P_i = P_{Gi} - P_{Li} = \sum_{j \in \mathcal{N}_i} P_{ij} = \sum_{j \in \mathcal{N}_i} \frac{1}{X_{ij}}(\theta_i - \theta_j) \tag{20}$$

We can express this in matrix form as $P = B\Theta$ where $P = \begin{bmatrix} P_1 & \cdots & P_N \end{bmatrix}^T$, $\Theta = \begin{bmatrix} \theta_1 & \cdots & \theta_N \end{bmatrix}^T$, and B is a "weighted" Laplacian matrix for the graph, G, whose ijth element is

$$B_i = \begin{cases} \sum_{j \in \mathcal{N}_i} \frac{1}{X_{ij}} & \text{if } i = j \\ -\frac{1}{X_{ij}} & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

Based on our DC power flow model, we can now formulate the power dispatch problem as follows. We assume there exists a function $U_i : \Re \to \Re$ for each $i = 1, \cdots, N$ which represents the "cost" associated with generating power at bus i. Note that we consider a very generalized notion of cost that would, in practice, be determined by economic considerations. But we can also treat it as a priority assignment that may be driven by formalisms that model the "consequences" of losing power with respect to a military mission (see Sandia Lab Reports by Menicucci and Akhil). In general, we assume that these "cost" functions of convex functions. The dispatch problem is formally stated below

$$\begin{array}{rc} \text{minimize:} & U(P_G) = \sum_{i=1}^{N} U_i(P_{G_i}) \\ \text{with respect to:} & P_G \\ \text{subject to:} & B\Theta = P_G - P_L \\ & \underline{P}_G \leq P_G \leq \overline{P}_G \\ & \underline{P} \leq A\Theta \leq \overline{P} \end{array} \tag{22}$$

In this case P_G is a vector of the generated power and P_L is a vector of the local loads on each bus. A is the graph's incidence matrix and B is the graph's weighted Laplacian matrix (that were defined above). $\underline{P}_G$ and $\overline{P}_G$ reprseent lower and upper limits on generated power. $\underline{P}$ and $\overline{P}$ represent lower and upper limits on the power flows in the transmission lines. This problem seeks to find the optimal generated power such that the total generation cost is minimized subject to the power flow equation and physical constraints on the generation and transmission systems.

Appendix B: Network Utility Maximization Problem [3, 7]

We're using a distributed approach to solve the dispatch problem. What this means is that a processor will be located at each generation point within the power system. These buses will communicate with each other over a wireless communication network and will use their local processor (computer) to make local decisions about optimally setting their generated power P_{Gi}. This local decision will be made on the basis of that bus' local "load" and power flows, as well as information receiving form "neighboring" buses. By "neighboring", we mean that buses that are directly connected to the bus through a single transmission line. The key point of the dispatch algorithm being developed is that all decisions setting a bus' generating power are made "locally". There is no "centralized" decision making taking place within the system.

The dispatch problem can be viewed as a variation on the network utility maximization (NUM) problem that has been recently used in congestion control over computer networks. The NUM problem assumes that we have N users trying to transmit their data over M links in a communication network. Figure 13 shows such a situation where we have 2 users transmitting over 4 links.

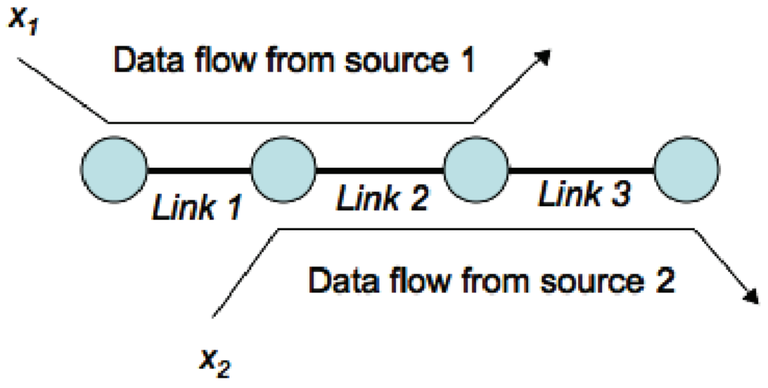


Figure 13: NUM Problem

The formal statement of the NUM problem is as follows

$$\begin{array}{rc} \text{maximize:} & U(x) = \sum_{i \in \mathcal{S}} U_i(x_i) \\ \text{subject to:} & Ax \leq c \text{ and } x \geq 0 \end{array} \tag{23}$$

In this case, x is a vector of the user transmission rates, A is an incidence matrix for the graph, and c is a vector of limits on the total data flowing through each link in the network. Rather than minimizing a cost, we seek to maximize a "utility" function U_i. As we can see there are some similarities between the NUM problem and the Dispatch problem. The dispatch problem seeks to minimize total cost subject to power flow constraints at the generators and transmission lines. The NUM problem seeks to maximize "utility" subject to flow constraints on the link.

Since these two problems are similar, it seems that we should be able to apply recent methods for solving such problems to the dispatch problem. In particular, recent work by Lapsley and Low have developed a dual-decomposition algorithm that solves the NUM problem in a completely distributed manner. This is done by first dualizing the original problem to obtain

$$\begin{array}{rc} \text{minimize:} & \max_{x \geq 0} \left[\sum_i U_i(x_i) - p^T(Ax - c) \right] \\ \text{subject to:} & p \geq 0 \end{array} \tag{24}$$

This transforms the original constrained optimization problem to an unconstrained problem. The new variable p is a Lagrange multiplier that has the "physical" interpretation of being a (shadow) price that the link charges users for using the link. Obtaining a solution to this dual problem also solves the original NUM problem. The key point is to find a "distributed" algorithm for solving this.

Dual decomposition is one such distributed algorithm for solving the NUM problem. The particular updates used by dual decomposition are

$$x_i[k+1] = \arg\max_{x_i} \left[U_i - x_i \sum_j p_j \right] \tag{25}$$

$$p_j[k+1] = \max \left\{ 0, p_j + \gamma \left(\sum_i x_i - c_j \right) \right\} \tag{26}$$

This update law is distributed in the sense that each user update only needs information from the links that are used by that user and each link (price) update only needs information from the users that are using it.

One of the key weaknesses of algorithms like dual-decomposition is that the step-size (γ) used in updating the above equations may need to be extremely small to ensure stability of the recursion. This has a significant consequence on the complexity of message passing over the radio network supporting the algorithm. In particular, the stabilizing step size has already been shown to be inversely proportional to some common measures of network complexity such as neighborhood size and path length. This bodes ill for using algorithms such as dual-decomposition to solve truly large-scale versions of the NUM problem.

Recently, we have found [8] that it is possible to greatly reduce the message passing complexity of algorithms such as dual-decomposition by using an "event-triggered" approach to message passing. The event-triggered approach broadcasts link/user information when "local errors" exceed a state-dependent threshold. This approach has been shown to reduce message passing by up to two orders of magnitude. This is shown in figure 14 where we plot the message complexity (number of transmitted messages) as a function of the maximum neighborhood size in the network. The results clearly show that algorithms such as dual-decomposition have a superlinear complexity whereas the complexity of a comparable event-triggered implementation is 1-2 orders of magnitude lower.

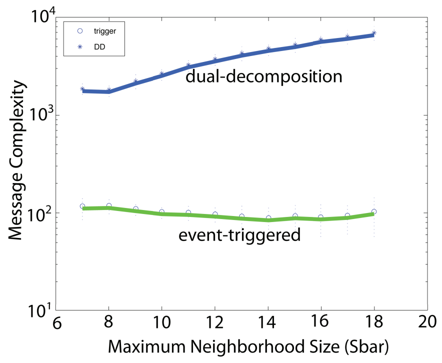


Figure 14: Message Complexity versus Neighborhood Size for Dual-Decomposition and Event-triggering

Appendix C: Event-Triggered Optimization [9]

In distributed decision systems, individual agents must exchange information to better coordinate their activities. This message passing is done over a digital communication network. While digital communication technology is extremely mature, the technology possesses fundamental physical limitations that can greatly impact the cost and performance of the networked system. Digital communication networks can only broadcast messages at discrete-time instants. These systems often use wireless radio frequency (RF) channel that must be shared amongst several users. The possible collisions of messages in this shared channel can greatly reduce system throughput and increase message latency. These considerations introduce a degree of unreliability that can greatly degrade the ability of intelligent agents to successfully coordinate their actions.

One way of addressing the reliability issue is to reduce the amount of communication required in support of the system application. This can be done through event-triggered message passing. Under event-triggering, each agent broadcasts its information to its neighbors when an internal error signal exceeds a state-dependent threshold.

To clearly explain how event-triggering functions, let's consider a collection of N dynamical agents. Sequences of *broadcast times*, $\{B_i[k]\}|_{k=0}^{infty}$ and *reception times*, $\{R_i[k]\}_{k=0}^{\infty}$ are associated with the ith agent. The time instants $B_i[k]$ and $R_i[k]$¡ represent the kth consecutive times when a message was broadcast or received, respectively, by agent i. The local state of the ith agent is a function, $x_i : \Re \rightarrow \Re^{n_i}$, where n_i is the dimension of the local state. At each broadcast time, the agent transmits its local state to its neighbors. That broadcast state is represented by the function $\hat{x}_i : \Re \rightarrow \Re^{n_i}$ where $\hat{x}_i(t) = x_i(B_i[k])$ for $t \in [B_i[k], B_i[k+1])$. This means, therefore, that $\hat{x}_i(t)$ is the local state of agent i at the most recent broadcast time.

In many applications, the ith agent's local state is a dynamic and may be assumed to satisfy the following piecewise continuous differential equation

$$\dot{x}_i(t) = f_i(x_i(t)) + \sum_{j \neq i} g_{ij}(\hat{x}_j(t)) \tag{27}$$

for all $t \in [R_i[k], R_i[k+1])$ and $k = 0, \cdots, \infty$. Equation 27 describes the evolution of the local state over the interval between two consecutive receptions of the ith agent. We usually choose the initial condition of this resulting state trajectory to yield a continuous trajectory when we join all of the segments together. The function $f_i : \Re^{n_i} \rightarrow \Re^{n_i}$ represents the ith agent's *local* dynamics. The function $g_{ij} : \Re^{n_j} \rightarrow \Re^{n_i}$ models the communication induced dynamic coupling between the jth and ith agents.

Equation 27 describes a *sampled-data* system in which the ith agent uses samples of its neighboring agent's local states to help regulate its own behavior. It is possible to characterize a broadcast time sequence $\{B_i[k]\}_{k=0}^{\infty}$ which guarantees that this sampled-data system is semiglobal asymptotically stable. This characterization of the stabilizing broadcast sequence is obtained by treating the sampled-data system as a "discrete" approximation to a continuously-sampled system that is already known to be input-to-state stable (ISS). The associated "continuously-sampled" system is described by the system equations

$$\dot{x}_i(t) = f_i(x_i(t)) + \sum_{j \neq i} g_{ij}(x_j(t)) \tag{28}$$

for all t. Note that equation 28 differs from equation 27 with regard to the argument of the coupling function, g_{ij}. In equation 28, the ith agent has *continuous* access to its neighbors' local state.

The asymptotic stability of the discretely-broadcast system is guaranteed if the continuously-broadcast system is input-to-state stable. In particular, this means there exists a function $V : \Re^n \rightarrow \Re$ (where $n = \sum_i n_i$) and class $\mathcal{K}$ functions (continuous and monotone increasing) $\alpha_i : \Re \rightarrow \Re$ and $\beta_i : \Re \rightarrow \Re$ such that

$$\dot{V} = \sum_{i=1}^{N} \frac{\partial V}{\partial x_i} \left(f_i(x_i) + \sum_{j \neq i} g_{ij}(x_j + e_j) \right) \leq \sum_{i=1}^{N} \left(-\alpha_i(\|x_i\|) + \beta_i(\|e_i\|) \right) \tag{29}$$

In the above equation e_i represents a disturbance that enters through the g_{ij} coupling function.

Equation 29 assures that the continuous-broadcast system (equation 28) is ISS with ISS-Lyapunov function V. This function will become Lyapunov function for the discrete-broacast system in equation 27 if it is possible to ensure that

$$-\alpha_i(\|x_i\|) + \beta_i(\|e_i\|) \leq 0 \tag{30}$$

for all time and all i. In particular, let the disturbance $e_i : \Re \rightarrow \Re^{n_i}$ be an error signal of the form

$$e_i(t) = \hat{x}_i(t) - x_i(t) \tag{31}$$

The signal $e_i(t)$ is therefore the difference between the ith agent's current state at time t and its state at the most recent broadcast time $B_i[k]$. At each broadcast time, $B_i[k]$, this error is zero. So one simply needs to choose the broadcast times $\{B_i[k]\}_{k=0}^{\infty}$ to ensure that equation 31 is always true. This can be accomplished by simply having the ith agent broadcast its local state when the inequality in equation 30 is about to be violated. In this regard, equation 30 is a threshold test automatically ensures that V is a Lyapunov function for the sampled-data system. So that selecting broadcast times based on equation 30 assures the asymptotic stability of the sampled-data system.

References

[1] K.E. Arzen. A simple event-based pid controller. In *Proceedings of the 14th IFAC World Congress*, 1999.

[2] A. R. Bergen and V. Vittal. *Power Systems Analysis.* Prentice-Hall, 2 edition, 2000.

[3] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[4] R. Lasseter. Control and design of microgrid components. Final Project Report - Power Systems Engineering Research Center (PSERC-06-03), 2006.

[5] R.H. Lasseter, A. Akhil, C. Mrnay, J. Stephens, J. Dagle, R. Guttronson, A. Meliopoulous, R. Yinger, and J. Eto. The certs microgrid concept. White paper for transmission reliability program, office of power technolgoies, U.S. Department of Energy, 2002.

[6] M.D. Lemmon, T. Chantem, X. Hu, and M. Zyskowski. On self-triggered full information h-infinity controllers. In *Hybrid Systems: computation and control*, 2007.

[7] S.H. Low and D.E. Lapsley. Optimization flow control, I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking (TON)*, 7(6):861–874, 1999.

[8] P. Wan and M.D. Lemmon. Distributed network utility maximization using event-triggered barrier methods. submitted to the European Control Conference, 2009.

[9] X. Wang and M.D. Lemmon. Self-triggered feedback control systems with finite-gain l2 stability. to appear in the IEEE Transactions on Automatic Control, 2008.