Integrated modeling of the electric grid, communications, and control

James Nutaro, Phani Teja Kuruganti, Mallikarjun Shankar
Oak Ridge National Laboratory,
Oak Ridge, Tennessee, USA
{nutarojj, kurugantipv, shankarm}@ornl.gov
Laurie Miller, Sara Mullen
University of Minnesota, Minneapolis,
Minnesota, USA
{mill0660, mull0197}@umn.edu
March 18, 2008

Abstract

This paper addresses a central concern in modeling and simulating electric grids and the information infrastructure that monitors and controls them. As the electric power infrastructure incorporates modern computing and communication technology, joint (hybrid) simulations of continuous power system models and discrete event models of communication, computation, and control operations become essential for understanding how the complete system behaves. In particular, new methods are needed to model and simulate hybrid scenarios for systems as large and complex as the electric grid. Here we offer a particular approach to modeling and simulation of hybrid systems as an enabling technology for analysis (via simulation) of modern electric power grids. Our approach, based on the Discrete Event System Specification (DEVS), integrates existing simulation tools into a unified simulation scheme. We demonstrate this approach with an integrated information and electric grid model of a distributed, automatic frequency maintenance activity.

1. Introduction

Modern communication technology has a prominent role in current efforts to improve the reliability and efficiency of electric power grids. Because of the central role that will be played by communication networks in the future power grid, accurate predictions of power grid behavior will be inseparable from accurate models of the underlying communication infrastructure. Accurate modeling and simulation tools for future power systems are needed to analyze electric power networks, digital communication networks, and their interdependencies.

Current approaches to hybrid system simulation do not readily permit the construction of very large hybrid system models. There are three major limiting factors, with different simulation systems possessing some or all of these traits. The first is a lack of expressiveness for discrete event dynamics. This is particularly true of continuous system modeling tools that require discrete events to be described in terms of event threshold functions; Beltrame describes several specific difficulties in this regard [4].

The second factor is lack of support for integrating external simulation models in a well-defined way. While almost every simulation environment allows for the

integration of external simulation software, few provide a rigorous definition of the resulting model dynamics [9]. This can result in simulated behaviors that are artifacts solely of the software integration. These artifacts can be difficult to anticipate and detect, making analysis of the simulation results an uncertain prospect.

The third factor is inadequate support for numerical integration of the continuous subsystems. This issue is manifested in discrete event simulation packages that have been augmented with continuous system simulation algorithms. The difficulty here is primarily technical, arising from the need to simulate continuous models that require advanced simulation algorithms (e.g., implicit integration schemes, special methods for stiff systems, and robust event detection schemes). Effective solutions can be constructed by first overcoming the issues described in the above paragraphs, and then defining an appropriate programming interface for continuous system simulation packages. Our proposed approach solves the first two problems, thereby establishing a necessary foundation for resolving this technical issue.

In the last decade, large cascading failures have raised concerns that the current control structure for the power grid is, or will be, unable to cope with an increasing demand for reliable electric power. Restructuring and deregulation in the electric power industry has fundamentally changed power transmission patterns, and new market pressures are eliminating capacity margins in both transmission and generation. As a result, electric power system operators are observing system behaviors that were uncommon in the previous decade, and for which there is limited operating experience [8].

Improving the reliability of electric power grids will require experience with uncommon system behaviors, broader situational awareness at regional control centers, fast and reliable automatic control beyond what the current protection system provides, and active participation by power consumers in the control process. It is broadly believed that transmission and generation shortfalls can be addressed in the short term by more effective use of existing resources (see, e.g., [16,8]). In the long term, elastic demand for electric power will improve the security of the power system [13]. This is motivating new and substantial interest in distributed power generation, distributed power storage, and demand response systems that rely on advanced communication technology (see, e.g., [23,26,15]).

Electric power grids contain a complex network of electromechanical control devices, sensors, and data communication systems. Power system emergencies are dealt with by two complementary systems. Automated protection systems act locally to quickly and automatically contain localized problems. Wide area control is applied by experienced system operators working in regional control centers. These centers use remote sensor data obtained through a wide area communication system. It is generally accepted that power systems in the future will incorporate data networks to enable fast, automatic, and coordinated response to contingencies.

A power system model that integrates generators, loads, transmission lines, control processes, and data networks is an example of a hybrid system. Physical laws that dictate the behavior of electromechanical subsystems are described by continuous equations, with simple discrete dynamics resulting from circuit breakers, relays, and other types of local response mechanisms. Data networks, on the other hand, are modeled as discrete event systems whose dynamic behavior is described by chains of significant events. Significant events in a communication model include the expiration of a timer, sending or receiving a packet, a packet buffer overflow, and

packet collisions on a shared bus.

There is little published work that describes techniques for simulating integrated models of power systems and network-based, wide area control schemes. Methods and tools described in the literature focus on integrating closed software packages [14,2]. These methods are driven by the need to integrate software, and the results often have limited application or make significant sacrifices of precision and accuracy. The scenarios that we consider for the electric grid require the systematic construction of non-trivial hybrid models; these models include complex continuous and discrete dynamics. The central goal of this paper is to describe a systematic approach to model and simulate the complicated interactions between the electric power and digital communication systems.

In Section 2 we give background and discuss related work that sets the stage for our proposed approach. We discuss the hybrid simulation challenges and also discuss the fundamentals of our hybrid modeling methodology. In Section 3 we map our hybrid modeling concept to an implementable formal structure defined in terms of the Discrete Event System Specification (DEVS). In Section 4 we discuss the implementation principles and a prototype simulation tool that we have constructed. Sections 5 and 6 illustrate our approach with a hybrid model of a smart grid system that aims to prevent under-frequency failures by using an automatic load control scheme.

2. Modeling Integrated Power, Control, and Communication Systems

The Open Access Same-Time Information System (OASIS) is a Web-based real-time bulletin board for monitoring available transmission capacity and coordinating power transfers. OASIS was mandated in the mid-1990s by the United States Federal Energy Regulatory Commission (FERC) for the North American power grid. OASIS operates over the Internet, using the Hyper Text Transport Protocol and IPv4, and it is one of the first examples of a network based monitoring and control system used by the United States power industry [20].

OASISNET is an interactive, Web-based simulation program that can be used to study the behavior of OASIS networks. It allows system operators to gain experience with OASIS through simulation [25]. OASISNET integrates a model of the OASIS application, its underlying communication protocol, and a load flow model of the U.S. electric power grid. OASISNET users can observe and effect simulated power flows through the simulated OASIS application.

A substation automation architecture described in [6] makes substantial use of standardized, packet based communication protocols. In the proposed system, intelligent electronic devices are integrated via commercially available Ethernet technology to create an integrated substation control system. Understanding, improving, and accurately predicting the performance of complex, IP based communication and control system will require detailed models of the complete, integrated system.

Futuristic scenarios, in which distributed software systems installed at generators and substations automatically detect and respond to problems that could result in power system failures, are being seriously considered by some power system research

organizations [18,3,17,12,5]. These proposals envision software embedded in every component of the power system to perform local monitoring and control. Embedded control software uses standardized and commercially available communication technologies to coordinate rapid control actions over wide geographic area.

These scenarios illustrate a growing need for integrated simulation of the continuous (analog) power systems and discrete (digital) communications and control. The EPOCH simulation environment [14] uses simulation middleware to integrate the PSCAD/EMTDC electromagnetic transient simulator, the PSLF electromechanical transient simulator, and the Network Simulator 2 (NS2) communication network simulator. This approach uses available power system and network simulation tools, and it thereby significantly reduces model development time. However, the simulation software for the power system model is not designed to handle external events. This forced the EPOCH designers to fix a synchronization interval for the two simulators, possibly introducing substantial errors into the hybrid system model. A similar methodology is used in [5] to integrate MATLAB Simulink and NS2.

Commercial and academic tools for building continuous systems simulations are widely available. For example, Modelica (http://www.modelica.org) is a standardized language for describing continuous systems, and several Modelica compilers are available. Several recent research teams have integrated some discrete event system modeling capabilities into existing continuous system simulation tools. A Modelica package for simulating statecharts is described in [10]. Mosterman et al [19] showed that classic Petri Nets can be simulated with Modelica by using time and state events. These efforts have met with considerable success, but they are limited by both simulation software performance and the range of discrete event systems that can be modeled.

The SimEvent package from Mathworks, currently being developed as part of the MATLAB Simulink environment, promises to overcome problems of expressiveness and efficiency by providing a combined tool for modeling continuous and discrete event systems. This package has become available only recently, and its internal implementation has not, as yet, been discussed in the open literature.

The solutions described above have a common drawback; they require complex discrete event simulations to be rebuilt from scratch for use with a particular continuous system simulation package. This presents a significant obstacle to studying systems with complex discrete dynamics. This is particularly true when attempting to build a model of a modern communication network.

A more viable approach is to introduce continuous models as components in a discrete event simulation. Several decades of research and commercial development of continuous system modeling languages have produced standard languages (such as Modelica) for describing continuous system models. Model reuse is one of the driving goals of these standards. Many continuous system models that are described in this way can be compiled into a form that is amenable to integration with discrete event simulation tools. At the same time, the simulation code that is produced by the compilers can use advanced numerical integration and event detection algorithms.

Our approach makes it possible to integrate mature continuous system simulation technologies with powerful discrete event simulation tools. Discrete event simulation packages that accurately model communication networks are widely available. Moreover, existing continuous simulation languages can, in principle, be modified so

that the software produced by their compilers provides hooks for integrating it with discrete event simulation tools. For example, the acslXtreme product, from AEgis Technologies, has a substantial programming interface to the simulation code produced by the ACSL compiler, and extensions to this API for the purpose of hybrid system simulation are being considered. In light of these facts, a reasonable path forward is to include continuous models in discrete event simulation frameworks; our work takes this approach.

3. A DEVS-based Approach to Hybrid Simulation

A general approach to integrating continuous models into a discrete event simulation framework is described in [27,11]. This approach can avoid problems of expressiveness and efficiency with respect to modeling and simulating discrete event systems. At the same time, powerful continuous system simulation algorithms can be easily incorporated into the proposed framework.

A hybrid system with discrete input and output dynamics can be described with four functions. The first function describes how the continuous, internal state variables evolve between discrete events. The second function indicates how much time will elapse before the next discrete internal event. The third function determines how the system output changes in conjunction with discrete internal events. The fourth function describes discrete state change in response to internal (state) and external (input) events.

The dynamics associated with this structure can be summarized as follows. The hybrid model is partitioned into a continuous part, possibly with state and time events, and a discrete event part. The continuous part is simulated using a continuous system simulation algorithm, but it schedules integration time steps using the discrete event simulator's event scheduling mechanism. The discrete event part is simulated without change. The continuous system simulator schedules internal (self) events as dictated by its numerical integration scheme and event detection algorithm. When self scheduled events occur the simulator updates its state variables and, when required, schedules events to trigger behaviors in the discrete event part of the system. External events are handled just as time events would be handled and a new internal event is scheduled based on the new system state. The success of this approach is predicated on properly partitioning the system into its discrete and continuous parts.

This notion of a hybrid system is formalized with a structure

$$
\begin{aligned}
&X \text{ and } Y \text{, the input and output value sets} \\
&S \text{, the internal state set} \\
&F : S \times R \to S \text{, the evolution function} \\
&G : S \to R \text{, the event scheduling function} \\
&A : S \times X_{\Phi} \to S \text{, the discrete action function,} \\
&\text{where } X_{\Phi} = X \cup \{\Phi\} \text{ and } \Phi \text{ is the non} - \text{event, and} \\
&L : S \to Y \text{, the discrete output function.}
\end{aligned}
\tag{1}
$$

The set X is the range of values that can be injected into the system. The set Y is the range of values that can be produced by the system. The set S is the range of the system's internal state variables.

The evolution function $F(q,h)$, with $q \in S$ and $h \in R$, takes the system from a state q at time t to a later state q' at time $t+h$. This function describes continuous, autonomous evolution of the model's internal state. The system evolves continuously until $G(q)=0$ or a change occurs in the input trajectory. At these points, the discrete action function dictates an immediate and, possibly, discontinuous state change.

The discrete action function $A(q,u)$, with $u \in X_\Phi$, determines the response of the model to discrete events. These events can be inputs to the system or be triggered by its internal dynamics. In either case, the system changes state instantaneously from q to $q' = A(q,u)$. Changes due to internal dynamics occur when $G(q)=0$, and the subsequent state of the system is determined by $A(q,\Phi)$. External events are due to a change in the input trajectory. In this case, the state immediately following the event is given by $A(q,x)$, where $x \in X$ is the value of the input trajectory immediately after the change occurs.

The discrete output function $L(q)$ defines the model's output trajectory. The initial output value is given by $L(q_0)$, where q_0 is the initial state of the system. Discrete changes in the output trajectory occur when $G(q)=0$. At these times, the output trajectory takes the value $L(q)$, and it keeps this value until the system again enters a state in which G evaluates to zero.

The dynamics associated with the structure (1) are defined by associating its elements with a DEVS atomic model. The atomic model state set is S, and its input and output sets are X and Y. The state transition, time advance, and output function of the atomic model are defined in terms of the evolution function F, event scheduling function G, discrete output function L, and discrete action function A. These definitions are

$$\begin{aligned}
&\delta_{int}(q) = A(F(q,ta(q)),\Phi) \\
&\delta_{ext}(q,e,x) = A(F(q,e),x) \\
&\delta_{con}(q,x) = A(F(q,ta(q)),x) \\
&ta(q) = G(q) \\
&\lambda(q) = L(F(q,ta(q)))
\end{aligned} \tag{2}$$

In this representation, output and internal events coincide with the expiration of the time advance. The discrete output is computed using the system state just prior to the discrete event (i.e., prior to applying the discrete action function).

An implementation of this system will, in general, require events that do not result in discrete actions (i.e., an evaluation of A) or discrete output (i.e., an evaluation of L). These types of events are needed, for instance, when the evolution and event scheduling functions are implemented with numerical integration and state event detection algorithms (see, e.g., [7]). A DEVS model that is functionally equivalent to (2) can be had by defining a function

$$IntegStep : S \to R$$

that picks the next integration step size. The system dynamics are then defined by

$$
\begin{aligned}
&\delta_{int}(q) = \begin{cases} A(F(q,ta(q)),\Phi) & \text{if } G(q) \le IntegStep(q) \\ F(q,ta(q)) & \text{otherwise} \end{cases} \\
&\delta_{ext}(q,e,x) = A(F(q,e),x) \\
&\delta_{con}(q,x) = A(F(q,ta(q)),x) \\
&ta(q) = \min\{G(q), IntegStep(q)\} \\
&\lambda(q) = \begin{cases} L(F(q,ta(q))) & \text{if } G(q) \le IntegStep(q) \\ \Phi & \text{otherwise} \end{cases}
\end{aligned}
\tag{3}
$$

4. Hybrid System Simulation with *adevs* and NS2

A hybrid simulation tool for network-based systems was developed using the NS2 [22] and *adevs* [1] simulation packages. Discrete event and continuous sub-processes that do not model communications are implemented using *adevs*. Communication specific processes are modeled using NS2. Models developed with *adevs* are integrated into the NS2 simulation model using the simulation control API that is part of the *adevs* simulation software. This API provides a collection for manipulating the underlying model as a single discrete event process (specifically, the resultant of the DEVS model; see [27]).

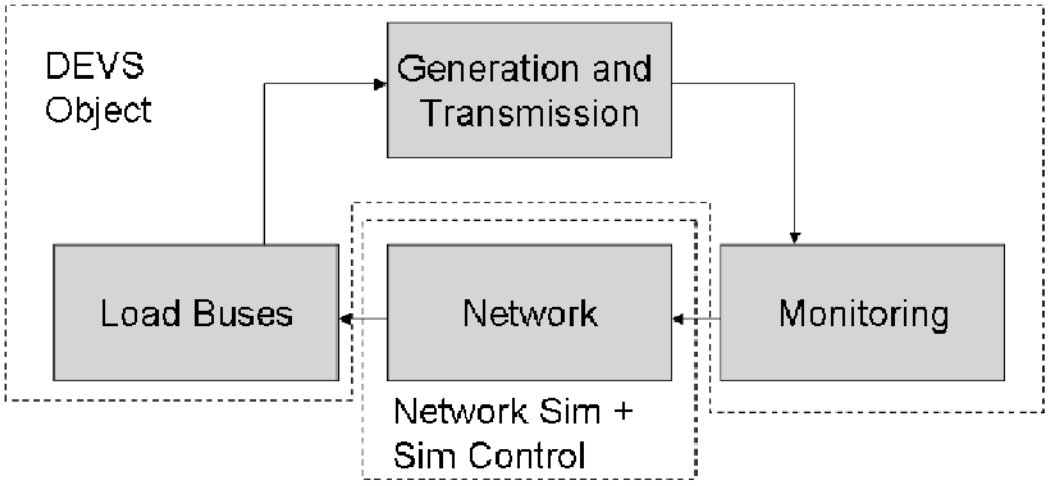

Figure 1. The simulation software architecture.

The *adevs* simulation software is encapsulated in an NS2 TclObject, and this TclObject is used directly by the NS2 simulation. The separation of the model into *adevs* and NS2 components is shown in Fig. 1. NS2 invokes the *adevs* components when two types of events occur:

1. an internal *adevs* event and
2. receipt of a message at a process modeled within *adevs*.

Type 1 events are scheduled by NS2 on behalf of the *adevs* simulator. One event of this type is scheduled at the beginning of the simulation. When a Type 1 event occurs, the NS2 simulator performs the following actions:

1. it queries the *adevs* module for any network messages that need to be sent,

2. it schedules NS2 events to send messages returned by the above query,
3. it tells the *adevs* component to update its internal state, and
4. it queries the *adevs* component for its next event time and schedules a corresponding *adevs* NS2 event.

Type 1 events cause the NS2 simulator to inject input into the *adevs* simulator in three steps:

1. inject input events into the *adevs* component,
2. tell the *adevs* component to update its internal state using the injected events, and
3. query the *adevs* component for its next event time and schedule a corresponding *adevs* event in NS2.

The DEVS structure (4) is used to simulate continuous processes. A fourth order Runge-Kutta integration scheme is used to solve the ODE set whose solution is the evolution function F. Zero crossing functions are used to describe the location of state events. Time events are scheduled explicitly.

State events are detected using discontinuity locking in conjunction with interval bisection. With this approach, the continuous variables are integrated over a single integration step. If the sign of a zero crossing function is unchanged at the end of the step, then it is assumed that no state event occurred in the interval. Otherwise, the integration step size is halved, and the test is repeated. If this procedure causes the step size to reach some small threshold value, then the event is assumed to occur at the end of that minimal step size. Our implementation uses the smallest single-precision floating point value (i.e., the C FLT_MIN macro) as this minimal time step value.

5. Example: Automatic Under-frequency Load Shedding

In this section, we use a notional system for automatic load control to demonstrate the capabilities of our hybrid simulation framework. The continuous component in this system is a power generation and transmission model of a 17 bus system that is based on the IEEE 14 bus system. To accommodate the power flow calculations for this model, three additional buses were added to the IEEE 14 bus model to separate buses with both load and generation. The model consists of twelve loads and five generators that are interconnected as shown in Fig. 2.

The power flow and load shedding aspects of this model have been simplified in two ways. First, the power system model was developed to study the mechanical response of generators to significant load changes and generator or transmission line losses [21]. The bus voltages are assumed to remain constant, line resistance and shunt reactance are neglected, and a linearized DC power flow is used to calculate the real power flows in the network. This simplification is justified because the load control mechanism studied here is triggered by frequency variations; consequently voltage deviation and reactive power flow are neglected. Second, all loads are assumed to have an interruptible component, and every load responds immediately to shedding requests. This simplifies both the controller and load model by assuming ideal load behaviors. We expect that a model which includes both reactive power flow and market driven load behaviors may predict different outcomes from what is shown

here. Nonetheless, these two simplifications give us a manageable starting point and allow for a clearer presentation of the model and its hybrid dynamics.

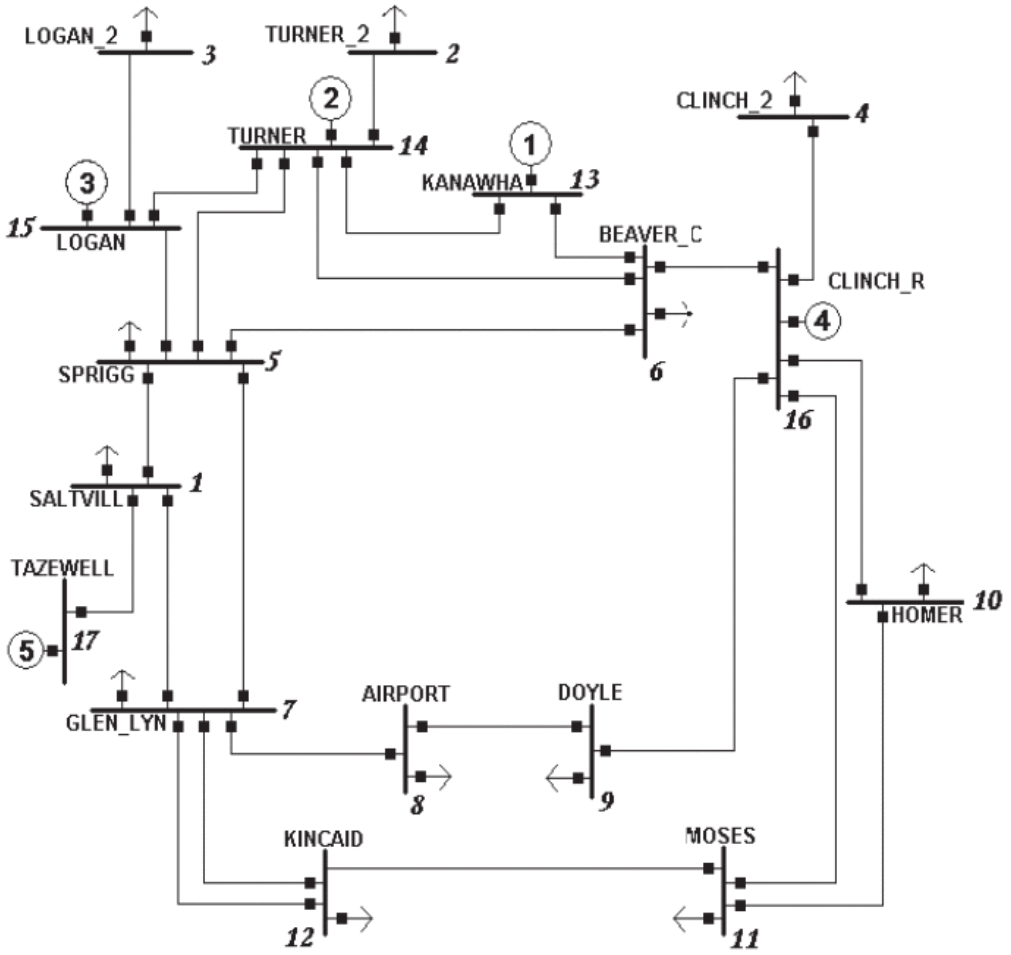

Figure 2. The 5 generator and 12 load bus power system model.

The generators are modeled as synchronous machines using the swing equation plus additional equations that model a governor, non-reheat turbine, and overspeed breaker. One of the five generators also includes a basic Automatic Generation Control (AGC) unit that eliminates steady-state frequency error throughout the system. The three equations that describe the generator dynamics are

$$\Delta\dot{\delta}_g = \Delta\omega$$
$$\Delta\dot{\omega} = (\Delta P_m - \Delta P_e)/M$$
$$\Delta\ddot{P}_m = -100(k_{agc}\Delta\delta_g + \Delta\omega/R + 0.25\Delta\dot{P}_m + P_m)$$

where ΔP_m and ΔP_e are the deviations of mechanical power output and electrical demand from the initial steady-state operating point P_0, $\Delta\delta_g$ is the change in the relative generator shaft angle, and $\Delta\omega$ is the deviation of the shaft angular velocity from 60Hz. The values R, M, and k_{agc} are the generator's speed droop constant, rotational inertia, and AGC gain. If the speed deviation of a machine exceeds ± 0.1 Hz, then it is disconnected from the transmission network.

Real power flow is calculated from generator shaft angles and electrical power

demand at the load buses. Disconnected generators are treated as load buses with zero power demand [21]. To facilitate the calculation of electrical demand P_e on the generators, the network admittance matrix Y is broken into the four sub-blocks shown in Eqn. 6. The Y_{ll} block describes load to load connections, Y_{lg} and Y_{gl} describe the symmetric generator-load/load-generator connections, and Y_{gg} describes generator to generator connections. Similarly, the bus angle and electric power vectors are split into upper and lower blocks. The vectors $\overline{\Theta}_l$ and $\overline{P}_l$ denote the load bus angles and injected power. The vectors $\overline{P}_e$ and $\overline{\Theta}_g$ are the electrical demand on the generators and the generator shaft angles. The power flow equations are

$$\begin{bmatrix} \overline{P}_l \\ \overline{P}_e \end{bmatrix} = \begin{bmatrix} Y_{ll} & Y_{lg} \\ Y_{gl} & Y_{gg} \end{bmatrix} \begin{bmatrix} \overline{\Theta}_l \\ \overline{\Theta}_g \end{bmatrix} \tag{6}$$

and the electrical demand on the generators is given by

$$\overline{\Theta}_l = Y_{ll}^{-1}(\overline{P}_l - Y_{lg}\overline{\Theta}_g)$$
$$\overline{P}_e = Y_{gl}\overline{\Theta}_l + Y_{gg}\overline{\Theta}_g .$$

Attached to each generator is a monitor that samples the generator state at a fixed rate of $1/T$. The sampled variables are the generator mechanical power ΔP_m, rate of change in mechanical power $\Delta \dot{P}_m$, electrical load ΔP_e, shaft velocity $\Delta\omega$, and shaft acceleration $\Delta\dot{\omega}$. At each sampling instant, the monitor estimates time to an under-frequency failure by

$$t_f = \begin{cases} \dfrac{60.0(\Delta\omega + 1) - 59.9}{|\Delta\dot{\omega}|} & \text{if } \Delta\dot{\omega} < 0 \\ \infty & \text{if } \Delta\dot{\omega} \geq 0 \end{cases}$$

and time to meet demand by

$$t_s = \frac{|\Delta P_m - \Delta P_e|}{|\Delta \dot{P}_m|}.$$

The generator is in danger of being disconnected if

$$t_f \leq t_s .$$

In this case, the monitor will broadcast a request asking all load buses to reduce their power demand. If, on the other hand,

$$t_f > kt_s$$

where $k >> 1$ is a safety factor, the system is operating under capacity. In this case, the monitor will generate a message indicating that electrical power demand can be increased.

Load change requests are summarized by the load service fraction α, with $\alpha \in [0,1]$. When $\alpha = 1$, the generator can tolerate the full electrical demand seen at its terminal. When $\alpha < 1$, the generator would like to see the demand on its terminal reduced to a fraction α of the full power demand. Changes to α occur in discrete increments $\Delta\alpha$

at a maximum rate of $1/T$.

The operation of the monitor is depicted graphically in Fig. 3. The monitor state and output are computed at each sampling time. Circles denote discrete phases, and the action performed in each phase is denoted by *state change / output*. Arrows denote phase change conditions. At each sampling instant, the phase change conditions are evaluated and the phase is changed accordingly. Then the output value is produced and, subsequently, the state variable change is applied. A new monitor state and output is calculated every T seconds using the sampled generator variable values.

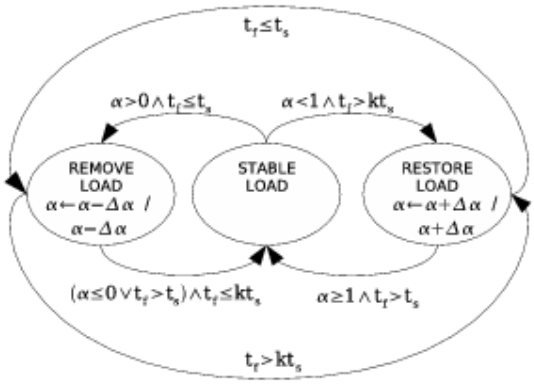

Figure 3. State transition diagram for the generator monitor.

Load buses remember the last load service fraction received from each monitor. The remembered requests are denoted α_i, with $i \in [1,5]$ indicating the monitor that produced the request. Each load bus is also aware of its electrical demand. On receiving a message, the load bus removes or restores some of its power demand, resulting in an overall reduction in demand on the generators. Each bus assumes that it produces $1/12$ of the total electrical demand, and that each generator carries $1/5$ of the total electrical demand. Therefore, whenever a single generator requests a reduction of its load from 1 to α, each load bus disconnects only $(1-\alpha)/60$ of its total electrical demand from the transmission network.

This reasoning leads to the load control rule that is implemented by each load bus. Letting L_d be the full power demand at a load bus, the amount of load served, denoted L_s, is equal to

$$L_s = L_d\left(1 - \frac{1}{12} + \frac{1}{60}\sum_{i=1}^{5}\alpha_i\right).$$

If the α_i are 1, then all of the demand at the load bus is met (i.e., $L_s = L_d$). Otherwise, each load bus sheds a fraction of its electrical demand. However, no individual load bus is willing to reduce its power demand by more than 8.3%, corresponding to all the α_i having a value of zero.

The monitors and load buses communicate through a packet switching network. Communication lines follow the network transmission lines, and packets are routed from origin to destination through this shared communication medium. The communication lines are modeled as queues with a fixed throughput, measured in bits per second (bps), and base delay. The time required for a packet to traverse a single line is given by

$$\frac{\text{bits}}{\text{throughput}} + \text{base delay}.$$

Each line has a buffer for queuing packets, and only one packet can traverse the communication line at any time. No packets are dropped. In general, a message will need to travel through several lines before reaching its destination. Network flooding is used to implement the broadcast function, with unique packet identifiers used to prevent re-broadcasting of packets that have already been processed [24].

6. EXPERIMENTS

Two sets of experiments were conducted to study how base line latency and throughput impact the effectiveness of the control scheme. A fixed parameter set was used for the generators, electrical transmission network, and monitors. The line admittances that were used are described in [21]. The initial power P_0 at each generator is calculated to ensure steady state at a selected bus angle [21]. Other generator parameter selections are listed in Table 1. The controller parameters were $T = 0.01$, $k = 10^4$, and $\Delta\alpha = 0.1$. The size of a control message was fixed at 900 bytes. Base link latency and throughput were varied independently. Network performance is summarized by the time average percentage of load served, i.e.,

$$\text{Performance} = \frac{1}{t_{end}} \int_0^{t_{end}} \frac{\text{power supply}}{\text{power demand}} dt$$

where t_{end} is the end of the experiment observation time (20 seconds in this case).

Generator	$1/R$	k_{agc}
1	300	0
2	225	200
3	300	0
4	300	0
5	225	0

Table 1. Generator parameters

Load bus	$t = 0.0$	$t = 1.0$	$t = 10.0$
1	0.0		
2	-0.217		
3	-0.942		
4	-0.112		
5	-0.478		
6	-0.076		
7	-0.295	0.0	-0.4
8	-0.09	0.0	-0.09
9	-0.035		
10	-0.061	0.0	-0.4
11	-0.135	0.0	-0.135
12	-0.149	0.0	-0.149

Table 2. Electrical demand schedule

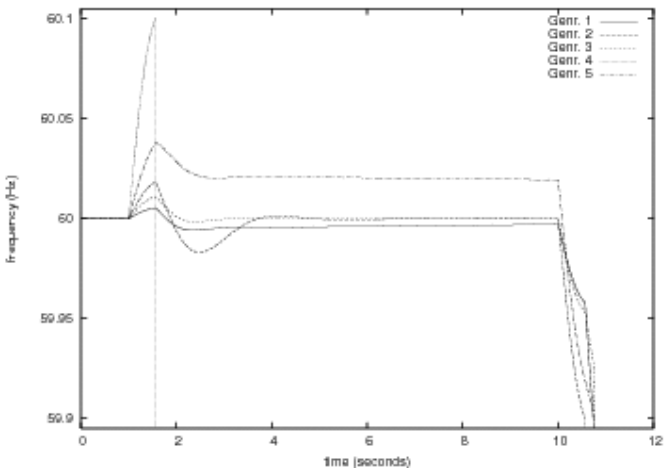

Figure 4. System failure in the absence of any load control scheme.

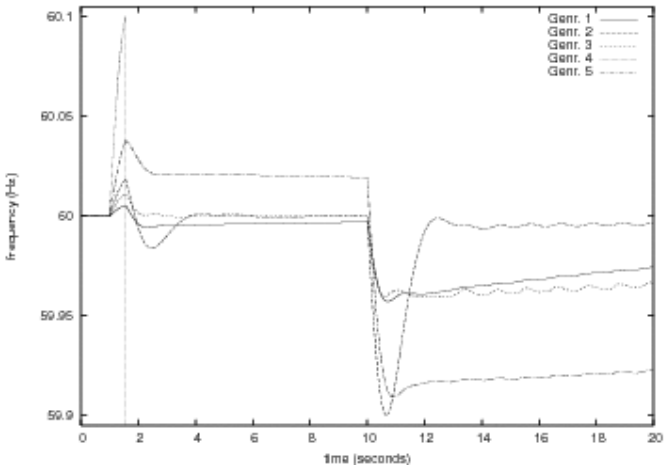

Figure 5. Generator frequencies when base line delay is 130ms and throughput is 1Mbps.

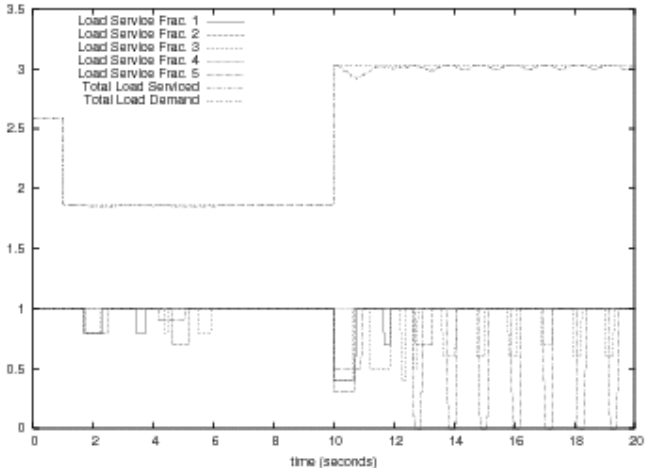

Figure 6. Total demand, served demand, and service fractions when base line delay is 130ms and throughput is 1Mbps.

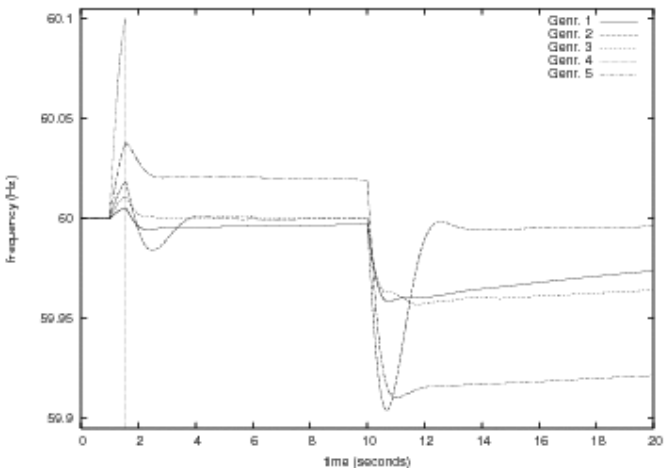

Figure 7. Generator frequencies when base line delay is 50ms and throughput is 1Mbps.

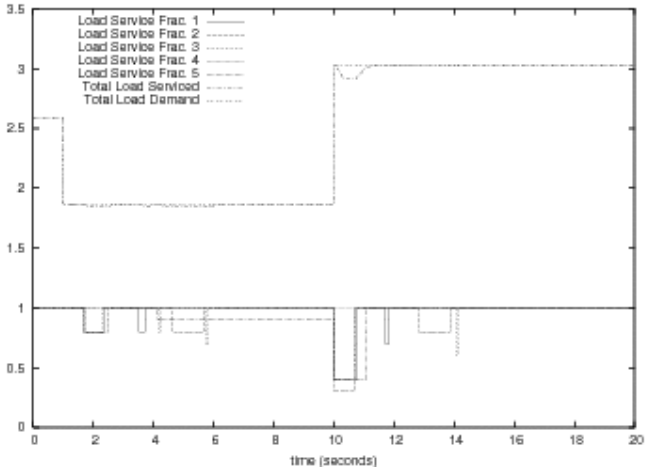

Figure 8. Total demand, served demand, and service fractions when base line delay is 50ms and throughput is 1Mbps.

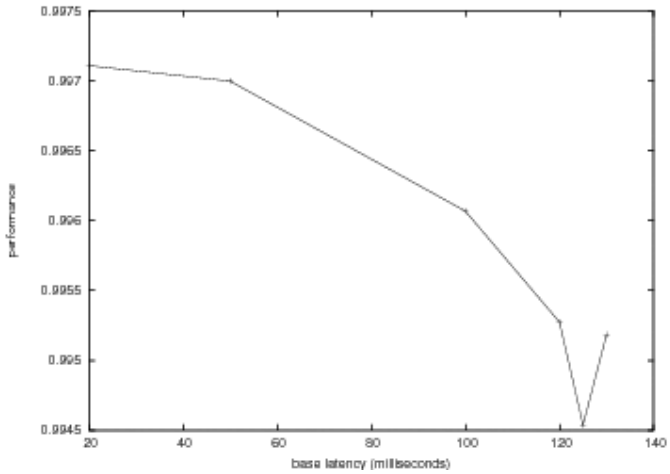

Figure 9. Performance as a function of latency with throughput fixed at 1Mbps.

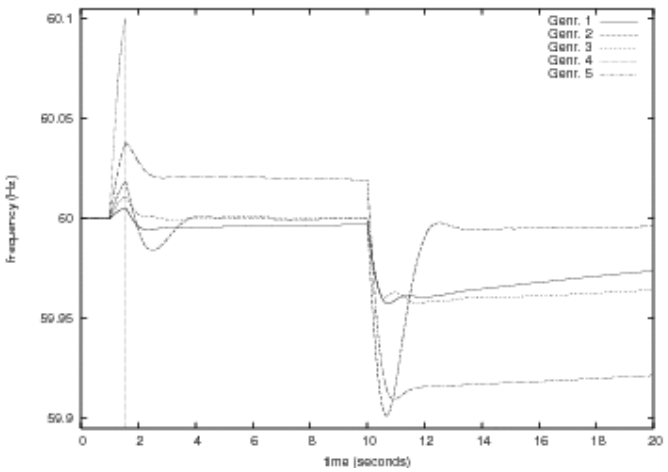

Figure 10. Generator frequencies when line throughput is 256Kbps and base latency is 20ms.

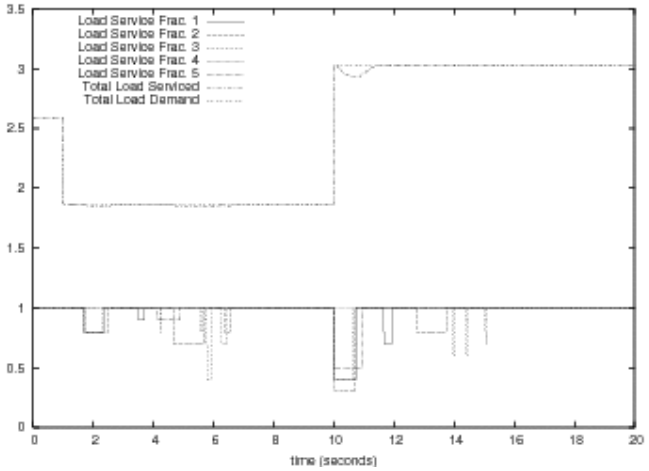

Figure 11. Total demand, served demand, and service fractions when line throughput is 256Kbps and base line delay is 20ms.

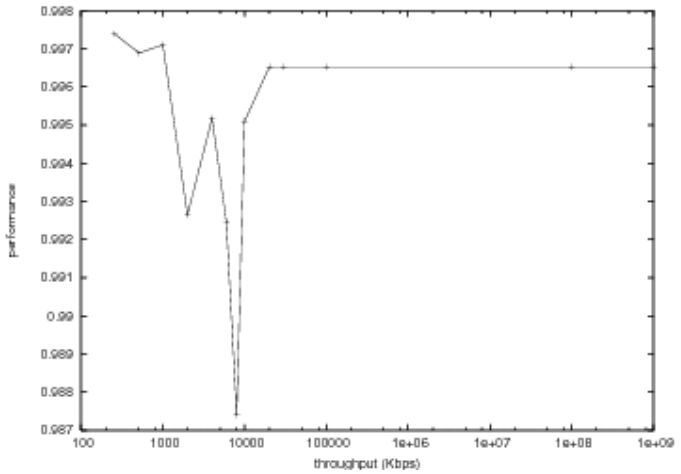

Figure 12. Performance as a function of throughput with base latency fixed at 20ms.

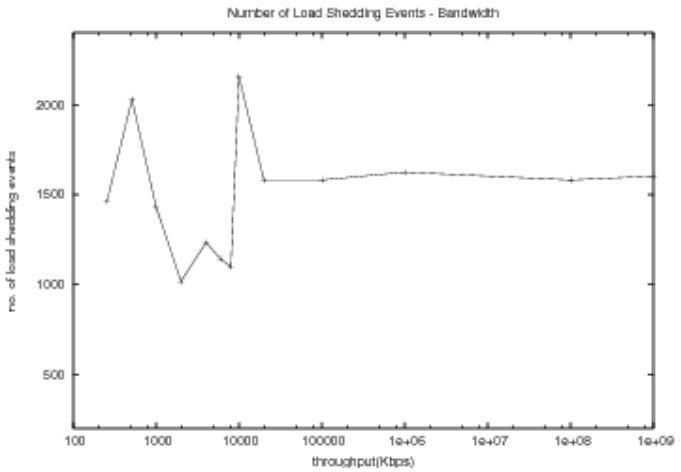

Figure 13. Number of load shedding events as a function of throughput with base latency fixed at 20ms.

The electrical demand schedule that was used in these experiments is shown in Table 2. The $t = 0$ column shows initial power demand at each bus. Subsequent columns contain an entry only for buses at which the power demand changes. Electrical demand is described by per unit power injected at the load bus. Without any load control, this schedule causes all five generators trip offline following the load spike at $t = 10$ seconds. The failure scenario is shown in Figure 4.

In the first set of experiments, the network throughput was fixed at 1Mbps, and the base line latency was varied between 20ms and 135ms. The five generators fail shortly after $t = 10$ in the 135ms scenario. The generator frequency, load service fractions, and electrical supply and demand totals for the 130ms scenario are shown in Fig. 5 and 6. This is the upper end of the survivable latency range, and there is a noticeable ripple in the system frequency that is caused by oscillations of the load service fraction at generators three and five. These ripples disappear and the system reaches a stable state at 50ms of latency, as shown in Figs. 7 and 8.

Figure 9 shows how the controller performance varies as a function of base line latency. The controller performance decreases gradually as the base latency increases. Catastrophic failure occurs when the base line latency reaches 135ms (not shown in the figure).

In the second set of experiments, the base line latency was fixed at 20ms and the throughput varied between 115Kbps and 10Gbps. The system fails at 115Kbps with all of the generators tripping offline. The generator frequency, load service fractions, and electrical supply and demand totals for the 256Kbps scenario are shown in Fig. 10 and 11. The system behaves nicely in this case, with the served load meeting demand approximately 5 seconds after the load spike at $t = 10$. The network performance as a function of throughput is shown in Fig. 12. The performance actually decreases as additional throughput becomes available, reaching minimum performance at 30Mbps, and then stabilizing near 99.65% at 100Mbps. Fig. 13 shows that the number of load shedding events changes as the line throughput changes. The number of load shedding events stabilizes as the performance metric stabilizes.

The surprising drop in performance as throughput increases is due to the complex interaction of the control scheme, electrical power flow, and communication network. As the link bandwidth changes, the queuing behavior in the network changes, and this

affects the order in which the loads are shed from the system. Because individual loads vary in size, the order in which they are reduced substantially impacts the behavior of the electrical network and load control system. Moreover, the impact of a particular load shedding event is dependent on when it occurs and where it is located in the network. The combined effect on overall system behavior would be difficult to anticipate without an integrated, dynamic model of the power, control, and communication systems.

7. Conclusions

Power grid modernization efforts need powerful modeling and simulation tools for hybrid systems. Improved situational awareness, substation automation, and distributed monitoring and control all presume the use of modern communication technology. The interaction of communication networks and the electric power system will be difficult to study without the aid of accurate models and efficient simulation environments.

Powerful modeling languages for continuous systems are already available. Models described in these languages can, in many cases, be compiled into a form that is amenable to integration with discrete event simulation tools. When compilers for these languages provide this capability it will be possible to build the complicated, large-scale hybrid models that are needed for engineering future power systems. In this paper we have shown that this can be done and demonstrated the key elements of our particular approach.

Modeling the continuous and discrete systems together, in a manner that preserves our formal definition of the hybrid dynamics, is the central contribution of this paper. We leveraged the DEVS methodology to create a hybrid model and simulate it using *adevs* and *NS2*. Our experiments with a load shedding scenario exposed behaviors that can only be observed with an integrated hybrid model. In particular, we find that the communication network affects the order of load shedding and that available bandwidth and network latency have a significant effect on the controller behavior. Identifying these behaviors requires detailed simulations. Because it is intractable to create analytical models of these integrated systems, simulation is necessary. Analysis of complex, integrated discrete and continuous processes requires accurate simulation techniques; our approach addresses this need.

References

[1] ADEVS: 2006, *adevs*: A Discrete Event system Simulator. Online at http://www.ornl.gov/1qn/adevs.

[2] Al-Hammouri, A., Agrawal, D., Liberatore, V., Al-Omari, H., Al-Qudah, Z. and Branicky, M. S.: 2007, Demo Abstract: A Co-Simulation Platform for Actuator Networks, Poster at the 5th ACM Conference on Embedded Networked Sensor Systems Conference.

[3] Amin, M. and Wollenberg, B.: 2005, Toward a smart grid: power delivery for the 21st century, *IEEE Power & Energy Magazine* (5), 34–41.

[4] Beltrame, T.: 2006, *Design and Development of a Dymola/Modelica Library for Discrete Event-oriented Systems Using DEVS Methodology*, Master's thesis,

Department of Computational Science, ETH Zurich, Zurich, Switzerland.

[5] Bhowmik, S., Tomsovic, K. and Bose, A.: 2004, Communication models for third party load frequency control, *IEEE Transactions on Power Systems* (1), 543–548.

[6] Caird, K.: 1997, Integrating substation automation, *IEEE Spectrum* (8), 64–69.

[7] Cellier, F. E. and Kofman, E.: 2006, *Continuous system simulation*, Springer.

[8] Cooke, D.: 2005, *Learning from the Blackouts: Transmission system security in competitive electricity markets*, Organisation for Economic Co-Operation and Development / International Energy Agency.

[9] Davis, P. and Anderson, R.: 2004, Improving the Composability of Department of Defense Models and Simulations, RAND Corporation.

[10] Ferreira, J. and de Oliveira, J.: 1999, Modeling hybrid systems using StateCharts and Modelica, *Proceedings of the 7th IEEE International Conference on Emerging Technologies and Factory Automation*, Barcelona, Spain.

[11] Giambiasi, N., Escude, B. and Ghosh, S.: 2000, GDEVS: A generalized discrete event specification for accurate modeling of dynamic systems, *Transactions of the Society for Computer Simulation International* (3), 120–134.

[12] Hauser, C., Bakken, D. and Bose, A.: 2005, A failure to communicate: Next-generation communication requirements, technologies, and architecture for the electric power grid, *IEEE Power & Energy Magazine* (2), 47–55.

[13] Hogan, W. W.: 1999, Market-Based Transmission Investments and Competitive Electricity Markets. Unpublished manuscript, available (as of Dec. 3, 2007) at http://ksghome.harvard.edu/ whogan/tran0899.pdf.

[14] Hopkinson, K., Wang, X., Giovanini, R., Thorp, J., Birman, K. and Coury, D.: 2006, EPOCHS: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components, *IEEE Transactions on Power Systems* (2), 548–558.

[15] Kiliccote, S., Piette, M. A., Watson, D. S. and Hughes, G.: 2006, Dynamic Controls for Energy Efficiency and Demand Response: Framework Concepts and a New Construction Case Study in New York, *Proceedings of the ACEEE 2006 Summer Study on Energy Efficiency in Buildings*.

[16] Kirby, B.: 2006, Demand Response for Power System Reliability: FAQ, *Technical Report ORNL/TM-2006/565*, Oak Ridge National Laboratory.

[17] Kuruganti, T., Shankar, M., Allgood, G. and Djouadi, S.: 2006, On Analysis of Control-Communication Integration in Distributed Power Grid's Electric, Information & Physical Domains, *Proceedings of the 16th Annual Joint ISA POWID/EPRI Control and Instrumentation Conference*, San Jose, California, USA.

[18] Massoud, A. S. and Wollenberg, B.: 2005, Toward a smart grid: power delivery for the 21st century, *IEEE Power and Energy Magazine* (5), 34–41.

[19] Mosterman, P., Otter, M. and Elmqvist, H.: 1998, Modeling Petri Nets as local constraint equations for hybrid systems using Modelica, *Proceedings of the 1998 Summer Computer Simulation Conference*, Reno, Nevada, USA, pp. 314–319.

[20] Mountford, J. and Austria, R.: 1999, Keeping the lights on, *IEEE Spectrum* (6), 34–39.

[21] Mullen, S.: 2006, *Power system simulator for smart grid development*, Master's thesis, University of Minnesota, Minneapolis, MN.

[22] NS2: 2006, The Network Simulator - ns2. Online at http://nsnam.isi.edu/nsnam/index.php/User_Information.

[23] Ropp, M., Larson, D., Meendering, S., Mcmahon, D., Ginn, J., Stevens, J., Bower, W., Gonzalez, S., Fennell, K. and Brusseau, L.: 2006, Discussion of a Power Line Carrier Communications-Based Anti-Islanding Scheme using a Commercial Automatic Meter Reading System, *Conference Record of the 2006 IEEE 4th World Conference on Photovoltaic Energy Conversion*, Vol. 2, pp. 2351–2354.

[24] Tanenbaum, A. S.: 1996, *Computer Networks, Third Edition*, Prentice Hall PTR, Upper Saddle River, New Jersey.

[25] Tian, Y. and Gross, G.: 1998, OASISNET: an OASIS network simulator, *IEEE Transactions on Power Systems* (4), 1251–1258.

[26] Watson, D., Piette, M., Sezgen, O. and Motegi, N.: 2004, Machine to Machine (M2M) Technology in Demand Responsive Commercial Buildings, *Proceedings of the ACEEE 2004 Summer Study on Energy Efficiency in Buildings.*

[27] Zeigler, B. P., Praehofer, H. and Kim, T. G.: 2000, *Theory of Modeling and Simulation, 2nd Edition*, Academic Press.