Implementing a Flexible Simulation of a Self Healing Smart Grid

Kendall E. Nygard, Steve Bou Ghosn, Davin Loegering, Md. Minhaz Chowdhury, Md. M. Khan, Ryan McCulloch, Anand Pandey
Department of Computer Science
North Dakota State University
Fargo, ND, USA

Prakash Ranganathan
Department of Electrical Engineering
University of North Dakota
Grand Forks, ND
prakashranganathan@mail.und.nodak.edu

Abstract—We describe the design and implementation of an agent based simulation for a smart grid. The design supports the evaluation of procedures for self healing.

The main design goal is to support the testing and comparing of alternative decision models that can specify grid-management actions that can be applied in scenarios that call for self healing. We first describe the agent and decision model design of our simulation. We then present the main features and innovations of our simulator, including such as representing the topology of the power grid, flexibility of the design, dynamic agent generation, scalability, and decision model independence. We also describe our work in progress.

Keywords- Multi-agent System; SmartGrid; Distributed computing; intelligent systems; Self-healing

I. INTRODUCTION

Agent-oriented software designs provide multiple advantages in Smart Grid simulations [1]. This type of design involves multiple interacting agents at supervisory and subservient levels, where each device, assessment, prediction, modeling, response, and decision-making procedure is associated with an agent. The agents are distributed, communicate and collaborate with each other, and take automatic actions to correct problematic conditions in the electrical grid. Our agent-oriented design is also scalable, with inherent provisions for handling distributed systems that are larger and more complex than those currently modeled.

There have been previous efforts to create simulation systems for a smart grid environment [2] [3][4]. In [2] the authors created a hardware simulation of a simple microgrid using MATLAB and Simulink to implement the functionality of low level electric circuits. They basically established the viability of an intelligent distributed autonomous system with autonomous agents that model a microgrid. Their agent implementation was very rudimentary, involving voltage monitoring to activate circuit breakers and to secure critical loads. The work was the advancing of a concept, with little agent interaction, analysis or evaluation. They did establish that a microgrid can be managed as part of the global grid and can work autonomously in islanded mode. In [3] an adaptive self-healing framework for power grids based on intelligent agent technologies is proposed, but the work did not into a working simulation. In [4], a promising flexible agent based simulation of a dynamic smart city is described.

The primary Smart Grid issues that can be addressed using the simulator are as follow:

- Which types of decision models are the most appropriate for handling self healing for specific alternative situations that can arise in a smart grid?
- Which types of decision models can maximize reliability and efficiency for a given power system?
- Which types of decisions can be made rapidly enough to avoid cascading failures, while still being deliberate enough to maintain the system efficiency?

Previous smart grid simulation research work falls short of providing satisfactory answers to these questions. Moreover, previous simulation designs have little flexibility to support to the simulating of different test cases. In particular, there is a strong need for simulation models to adequately handle dynamic as well as static topologies.

Our simulation design is highly flexible and scalable. It supports the creation new grid topologies from primitive elements and intelligent agent mappings through a convenient and easy to use graphical interface. The simulation can be manipulated at run time through various actions, including the shutting down of devices or the forcing of component failures. Finally, and importantly, the type of decision model used by the system agents can be configured, and entirely new models can be developed, run, and tested in the simulator. We see considerable potential for the simulations that we describe to help in the building a smarter electrical Grid architecture.

This paper is divided as follows: section II discusses the agent model and design used to implement our MAS (multi-agent system). Section III explains a general overview of how decision models are used and implemented in our simulation. Section IV discusses the main features and innovations of the simulator, gives an overview of its use, and describes potential benefits in Smart Grid research.
In this section we describe the agent design used in the implementation of our simulation. We use a three layer design, in which the two upper layers are agent based. The bottom layer (the physical layer) is a simple hardware simulation. Our aim with the bottom layer is to mimic the behavior of the electrical components themselves. Components modeled at this layer include relays, transformers, capacitors, power lines, consumers and generators, all running as autonomous units with no intervention or added intelligence. This separation between the intelligent agent layer and the physical simulation allows scenarios to run at the hardware level like the grid normally operates today, and, using intelligence agent support, to run with intelligent management.

At the physical layer we do not model full detail such as the modeling described in [2]. As an alternative, we treat the physical components as abstract black boxes and concern ourselves only with the basic power, voltage, and current metrics that are input and output to and from devices. This is because our focus is on the agent-based upper layers where our concern with how agents can solve fault situations, such as power outages, by assuming specific roles and collaborating with one another. For these kinds of purposes, knowing the details of the fault on an individual electrical component basis is not necessary in evaluating strategies for handling the fault by employing the decision models that generate the strategies.

In the middle layer of our design reside the supervised agents, including the Distributed Energy Resources (DER) agents, User agents, Device agents, and Control Agents. These agents collaborate with each other to achieve global goals, and are described below.

**User agents**: These agents act on behalf of consumers. For example, if there is an outage they engage in a negotiation with DER agents to obtain a power lease to address the outage.

**DER agents**: These agents act on behalf of power generation sources that are independent of the primary power distribution circuits. These sources are typically small companies or special consumers that also participate in the power market. Examples include wind power generation facilities, or geothermal generators, etc. DER agents act on behalf of their generators by engaging in power supply negotiations with users.

**Device Agents**: These agents are in charge of performing operations on devices. Examples include rerouting power by changing the distribution of power sent through the lines of a relay, disabling a certain power line on that same relay, shutting down a transformer or closing a circuit breaker to secure a critical load or island on some portion of the grid.

**Control agents**: These agents constantly monitor a section of the grid, collecting data and information from the devices and power lines in that geographical area. These results are sent in real time to a fusion point and aggregated. The results after fusion represent the status of the system at a certain moment in time. That fusion point where all data is received and aggregated is in the management agent, which is described below.

Middleware is required for the agents in the middle layer to communicate with the physical components in the first layer. This is achieved by encapsulating the bottom layer within an agent called the environment agent. The environment agent contains all information about the grid topology and all device status information at any given time. Middle layer agents query the environment agent to acquire information about the physical grid. They also send write/modify requests to the environment agent when they seek to alter the behavior of the physical grid. This simulates at an abstract level how the agents would be integrated with smart meters and other sensors in a real grid.

The top layer is administrative and hosts the management agents that make the high level decisions. The control agents send all data representing the status of the system at a given time to the management agent. Having received the relevant data, the management agents organize and analyze the data and detect situations in the grid that require self healing. When such a situation is detected, it creates a strategy to heal the system. This strategy is expressed as a set of corrective behaviors that prescribe a set of roles to be performed by the middle layer agents. Like the environment agent, the management agent has knowledge of the microgrid topology. It knows, for example, how components are physically connected, but it does not know if a line has failed or its current voltage level value. That information is sent in real time to the management agent by the control agents. The management agent is the primary decision maker in the simulation. However, the other agents in the middle layer still have certain autonomy to carry out the high level decisions generated by the management agent. Middle layer agents must make the specific decisions needed to carry out the higher level actions. Figure 1 illustrates the layered structure.
Figure 2 shows the communication flows between layers and illustrates the kinds of intelligence that is integrated into each layer. The management agent is on top and on the bottom we have the basic physical components simulated in the environment agent with no intelligence. The middle layer agents have some intelligence but they are limited in comparison with the management agent.

The framework used to implement our MAS is the Java Agent Development Framework (JADE). The primary reason we chose JADE is due to its strong compliance with agent communication standards like the IEEE standard for the Foundation for Intelligent Physical Agents (FIPA). This ensures portability and interoperability between many different systems and platforms. Other reasons are its convenience and ease of use. JADE runs agents in containers, which are agent groupings, and related agents usually run in the same container.

Figure 2. Flow of Communication between agents in our MAS

III. Decision Models Implementation Design

Decision models are at the core of our simulation, and as such there are many levels of decisions that can be taken by agents. The highest level decisions are made exclusively by the management agents. Management agents can use different decision models depending on the testing configuration. For example, it is possible to test a certain scenario using an Integer Linear Programming (ILP) decision model and then compare it against a Naive Bayesian decision model.

The decision process works in the following way: First, the management agent receives all relevant data representing the status of the system at a given time. Within the decision framework there is a special layer called the data presentation layer that is in charge of receiving the data collected from the control agents, and structuring it to make it compatible with the selected decision model. A decision model based on Bayesian logic might need the information presented in a different way than a one based on an ILP. If a researcher creates a new decision model and wants to integrate it into the simulator he/she must usually also create a data presenter that is appropriate for that model. An exception to this case is when one of the default data presenters already available in the simulator would work. In this way the data presenter layer of the decision model can be compared to the data presentation section of the application layer in the popular ISO networking model.

Once the Management agent receives the formatted data, it determines if the status of the power grid is normal or there is a need for self healing. In the latter case, it uses the decision model to generate a strategy that identifies corrective actions that can be taken to solve the detected problem in a high performance way. The decision model translates the results obtained into a set of corrective behaviors consisting of mappings of various middle layer agents to appropriate roles. The final output that the decision model generates is a set of actions and the agent responsible for accomplishing them, in the format that our MAS understands. The layer within the decision model that is in charge of converting the generated results to this format is called the Results Translator.

An example of a set of corrective behaviors could be:

- **DeviceAgent1** Set Line 2 in Relay +200Watts
- **DeviceAgent2** Close Circuit
- **DeviceAgent8** Shutdown Device
- **DERAgent6** Generate +100Watts

IV. Simulation Features And Design

In this section we discuss the main features of our simulation and explain the innovations that our simulator introduces when compared with previous smart grid simulations. We also provide a tour of the simulator, its basic configuration and use, and illustrate the potential benefits it brings to researchers testing self healing scenarios for a smart grid.

First we describe the design and run modes in which the simulator operates. In design mode, the simulator is configured with the details of a desired test scenario. At a minimum, designing a scenario requires inputting or loading a microgrid topology and the agent assignments to different components or sections of the microgrid. From this mode the user can save the design information to an XML file for later use. This makes it convenient for others to verify results or to run other tests on the exact same topology. In run mode, a test case created in design mode is loaded and then a simulation run is made. As the simulation proceeds the physical
infrastructure is emulated and the agents are active, monitoring and collaborating.

The main strengths of our simulation are as given below.
- Physical Layer Independence
- Power Grid Topology Design Flexibility
- Scalability
- Flexible Agent Mapping
- Dynamic Agent Generation
- Decision model Independence
- Active Control of the Testing Environment.

We describe of these strengths in turn.

**Physical Layer Independence.**

The physical hardware simulation is in a different layer than the agents, which allows considerable independence between them. Separating them allows us to run the simulator as a purely hardware simulation of the physical grid with no external intervention or intelligence. It is then possible to run the same configuration using agents and intelligence, and then assess how the control exercised by the agents improves the efficiency of the self-healing process of the power grid.

**Topology Design Flexibility.**

A primary goal in the simulation was to design it to provide flexibility to allow researchers to test a wide variety of grid topologies. We adopt a node/arc representation. The arcs model power lines and the nodes can represent consumers, generators or devices (like relays or transformers). This design allows researchers to generate a large variety of topologies.

Topology design of a microgrid is done in design mode using GUI tools, as illustrated in Figure 4. A toolbar along the left side of the window allows the user to drag and drop a variety of nodes onto the main grid layout area and connect them with power lines. The nodes can model consumers, generators, or devices and components like relays or transformers. The simulator has the flexibility to easily incorporate new types of components into the system as required. After a microgrid topology is designed, all the topology design information can be saved to an XML file for later use. The XML dialect is illustrated in Figure 5.

**Scalability.**

Design choices at the microgrid level provide for easy scaling of the system. Multiple microgrids can be interconnected to create larger microgrids and large scale grids. Within multiple microgrids a user can specify the positions of the internal and external connection points. Each of the microgrids in the simulator runs in an independent JADE container.

Once a user has generated several microgrids and established the connections between them, there is great flexibility under the JADE platform for executing the simulations over the microgrids. The constituent microgrids can be set to run on the same computer or distributed computer. Figure 6 illustrates multiple interconnected microgrids. The only limits to the scalability of the system are the processing power and related resources available to the user. The simulator itself has no specific limits, since different instances of it can be run to support each microgrid, and interconnected microgrids can communicate and interact with each other.
After a user has designed the topology of their grid and specified the design parameters, he/she can design and develop the desired agent support. The toolbar can be used to place agents within the network. A user can specify that available devices will be monitored or manipulated by agents by selecting the agent tool in the toolbar and clicking on the desired component to associate the agent with the device. Associations between agents certain consumers or generators are carried out with mappings in the same way.

As an example of the flexibility of this control, a user could have an agent assigned to monitor a certain device in one run, and then retest the same scenario without the agent monitoring that same device, to evaluate how that particular change affects the results.

One of the main features of our simulation is that the agents are dynamically generated in the supporting agent platform (JADE). When the user shifts the simulation from design to run mode the system will start the simulation of the physical grid and will automatically launch the JADE framework, dynamically creating the agents that the user mapped. The system is pre-configured to determine which kind of agent is required for each node. If, for example, the agent is placed on a consumer for which it knows it should create a user agent, on a generator a DER agent, and on a device a device agent, etc. Agent associations are illustrated in Figure 6. The agent-component mappings are saved to the XML file that stores design information.

In the case of a control agent, since it monitors a complete section or area in the grid instead of a single component, the set of components it monitors is determined by associating it with the nearest power line and then applying a hop-range parameter that can be set by the user, to decide the range within which components are monitored by that control agent. Only components within a hop-range number of hops from the original power line are monitored.

Every microgrid created in the simulator runs in a separate JADE container. Figure 7 provides an illustration. Agents that the user specified in design mode are created dynamically and placed in the microgrid JADE container and the system automatically creates an environment agent for the microgrid and at least one management agent.

A major innovation of our simulation system when compared to previous efforts is that of having the decision models independent within the design. Multiple customized decision models can be configured in the simulator. For example a management agent could be configured to use ILP (Integer Linear Programming) as its decision model to test a particular scenario and evaluate the efficiency and performance of how agents respond when using that decision model, and then test the same situation by configuring a heuristic decision model or one based on a Naïve Bayesian method, then compare the results. We anticipate the development of many decision models that will run seamlessly and be evaluated in the simulator.

Standard interfaces will allow researchers to design their own models and smoothly plug them in, without the need to know the details of the internal implementation of the simulator. An ILP model will be the first decision model to be integrated and evaluated by the [5].

The decision models for lower level agents are also configurable. For example DER and user agents use a...
prototype power negotiation model to establish power leases. That model is configurable with a simple default model that is based on the lowest price. More complex power negotiation models are under development.

Active Control of the Testing Environment.

The simulator allows users to configure parameters and settings when designing a physical microgrid and to change those parameters dynamically during run mode. An illustration of run mode is in Figure 8. Some examples of these parameters are the maximum power a given generator can produce the power demands of consumers, and the capacities of power lines. Parameters can be set and modified at both design and run time. For example, in design mode users can set an initial power/voltage/current that a component can output when the simulation starts, but in run mode it is possible to modify and reconfigure the voltage/power/current settings of a component in real time. Parameters set at design mode are saved with the rest of the configuration. When in run mode, a user can also generate different failures in the grid, such as a power line failure or a relay to malfunction. These actions can be used to observe the resulting effect and in particular the self-healing operations taken by whatever decision model is currently configured.

![Figure 8 Simulation in run mode](image)

V. CONCLUSIONS AND FUTURE WORK

The implemented simulation system satisfies the requirements of flexibility and scalability described in the design guidelines in [1]. It provides a powerful and versatile testing environment that allows the use of different testing configurations for every run, and gives the user control to manipulate the environment to determine how the agents would react to that particular situation. Furthermore, an important feature is that it allows users to choose which decision model should be used and to compare how agents behave differently in different situations using different decision models. Researchers are also able to design their own decision models and plug them into the system for testing.

Our work in progress includes heuristic decision models and the ILP model described in [5]. After these models are fully implemented, we will run them and evaluate their performance in the simulator and compare their performance and efficiency for alternative power grid self-healing scenarios.

VI. REFERENCES


