Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

EPOCHS: INTEGRATED COTS SOFTWARE FOR AGENT-BASED ELECTRIC POWER AND COMMUNICATION SIMULATION

Kenneth M. Hopkinson

Computer Science Department
Cornell University
Ithaca, NY 14853

Renan Giovanini

Department of Electrical Engineering
School of Engineering at São Carlos
University of São Paulo
São Carlos, SP 13566-590, Brazil

Xiaoru Wang

School of Electrical Engineering
Cornell University
Ithaca, NY 14853

Kenneth P. Birman

Computer Science Department
Cornell University
Ithaca, NY 14853

James S. Thorp

School of Electrical Engineering
Cornell University
Ithaca, NY 14853

Denis Coury

Department of Electrical Engineering
School of Engineering at São Carlos
University of São Paulo
São Carlos, SP 13566-590, Brazil

ABSTRACT

This paper reports on the development of EPOCHS, the **E**lectric **Po**wer and **C**ommunication Sync**h**ronizing **S**imulator, a distributed simulation environment. Existing electric power simulation tools do a good job of modeling power systems of the past, which were controlled as large regional power pools without significant communication elements. As power systems increasingly turn to protection and control systems that make use of computer networks, existing power simulators are less and less capable of predicting the likely behavior of large power grids. Similarly, the tools used to evaluate new communication protocols and systems have been developed without any attention to the roles they might play in power scenarios. EPOCHS utilizes multiple research and commercial off-the-shelf (COTS) systems to bridge the gap. EPOCHS is also notable for allowing users to transparently encapsulate complex system behavior that bridges multiple domains through the use of a simple agent-based framework.

1 INTRODUCTION

This paper presents EPOCHS, the **E**lectric **Po**wer and **C**ommunication Sync**h**ronizing **S**imulator, a combined simulation system, or federation, that links the PSCAD/EMTDC electromagnetic transient simulator, the PSLF electromechanical transient simulation engine, and the Network Simulator 2 (NS2) communication simulator. Each of these simulators is in some sense the best within its class. PSCAD provides extremely detailed simulations of power systems containing up to several hundred buses, and has been extensively validated through experiments comparing the behavior of PSCAD simulations with the behavior of real power systems. PSLF is used by electric utilities to simulate real-world situations and has undergone extensive validation. NS2 is the most widely used and trusted simulator for the Internet, and includes very high quality simulations of standard protocols like TCP and of standard Internet topologies, such as transit-stub configurations. Interesting issues arose during the creation of EPOCHS because none of its components were designed for simulation interoperability.

Using an approach like the DOD's High Level Architecture (HLA) (Kuhl, 1999) to link disparate simulation engines has many benefits. We live in an increasingly interconnected world where multiple domains such as water, power, and network communication can each affect the other. Yet, few standalone simulators exist that capture these inter-domain worlds. Constructing a combined simulation engine is oftentimes inordinately time-consuming and expensive. This is particularly true when simulations have both continuous and discrete-event components. An alternative is to link multiple simulations (federates) into a distributed environment (federation). Using this approach to combining multiple simulators for use in inter-domain situations is becoming more common. Commercial off-the-shelf simulation (COTS) systems are popular in many fields due to their rich feature sets, ease of use, and cost effectiveness. However, the lack of source code availability can be a hurdle to their use in federated simulation systems. This can be a significant handicap in areas where the most trusted systems are COTS software packages and built-in support for federating these simulators has not been implemented. Notable case studies federating COTS simulation software in an HLA or similar environment in the areas of supply chain manufacturing and manufacturing simulation have been performed in the GRIDS project (Taylor, 2001), in (Tucci, 2001)'s HLA compliance project, and in Strabburger's SLX federation (Strabburger, 1999), but the documented use of commercial simulation systems in federations is still relatively rare. Additionally, once a group of simulators have been federated, casual modelers may find the added complexity of the new simulation platform difficult to manage. These issues were foremost in our thoughts during the development of the EPOCHS system.

The technology underlying our work illustrates how non-intrusive techniques can be used to federate simulation engines using only the built-in Application Programming Interfaces (APIs). In addition, our agent-based framework hides the complexity involved in the combined simulation system making it easy for users to design new power scenarios involving communication. We believe that EPOCHS serves as an important case that could be applied to many settings such as air traffic control, banking, medical systems, military command and control systems, and other forms of mixed-mode critical infrastructure.

This paper is structured as follows. In section 2, we review background material that motivated the creation of EPOCHS and present the system's architecture. In section 3, we outline our agent framework. The methods used to federate the commercial and high quality research simulation components are discussed in section 4. We go on to describe a case study that has been developed and run on the EPOCHS platform in section 5. Finally, the paper ends with our conclusions in section 6.

2 EPOCHS

2.1 Motivation

An accurate understanding of an electric power system is necessary to have any confidence that it will operate reliably under the conditions that may arise in the field. Yet, electric power simulators today do not model the network communication that is being increasingly deployed in modern protection and control systems.

Traditional protection systems nearly always make decisions based on local measurements, and conventional control systems make use of slow communication systems that operated in a predictable manner. It has not been necessary to simulate communication in order to accurately model these electric power systems. However, over the last decade, power systems have been operated ever closer to their transmission, generation, and stability limits. Protection and control systems are being placed under a correspondingly greater strain. Power engineers have begun to conclude that the use of communication networks based on Internet standards is a natural choice to improve both. Yet, communication protocols and standards developed by the networking community were never thought to be used in power scenarios. There is a real concern that new protection and control systems will be deployed without fully understanding the issues that will arise from their use.

New kinds of simulation and evaluation tools are needed that are capable of bridging the gap, by providing high-quality simulations of electric power scenarios while simultaneously modeling the ways that computer communications protocols behave in realistic networks, confronted with realistic scenarios, including load surges, outages, and other forms of dynamic stress.

2.2 Overview

EPOCHS is a distributed simulation platform that links commercial and high quality research off-the-shelf simulators through the use of a Runtime Infrastructure (RTI) to allow modelers to investigate electric power scenarios that involve network communication. One goal of the EPOCHS platform is to minimize the intrusiveness of the simulation platform on users that are unfamiliar with its components. It does so by allowing complex behavior to be embedded within agents that can read and modify simulation variables by interacting with a module called the Agent Headquarters, or AgentHQ, that hides the details in the other simulation components. In the case of electric power systems involving network communication, this agent framework is a quite natural one since hardware support for software agents exists in the power system today.

EPOCHS seamlessly links its three off-the-shelf simulation systems from a modeler's perspective, enabling them to investigate power protection and control scenarios that combine communication with real-time sensing of the state of a power grid and real-time response. For example, suppose that a new protection protocol were deployed in a power system, and was known to operate correctly provided that the input data used by the control algorithm was accurate. EPOCHS will let us understand how that protocol might behave when running over TCP while other users move large data files through shared network links and routers – behaviors known to trigger delays and congestion control in the TCP protocol. Similarly, we are able to use EPOCHS to compare different options for running the same protection protocol (e.g. over TCP, over UDP, over various QoS mechanisms), and to understand how failures might impact power systems protection and control.

Broadly, EPOCHS is particularly valuable for evaluating the communications requirements of new protection and control schemes and the impact of common Internet behavior, such as TCP congestion control, on power system operation. Such information will help designers of future power systems' communications networks make appropriate design decisions, and will help designers of new control and protection protocols deploy them with confidence.

2.3 Architecture

Recent work centering on combining, or federating, simulation systems has focused on the use of the High Level Architecture (HLA). HLA is an architecture that can be used to combine individual simulations, known as federates, together into combined simulators known as federations. The "glue" that holds these combinations together is a central component known as a Runtime Infrastructure (RTI). The RTI routes all messages between simulation components and is responsible for making sure that simulation time is appropriately synchronized. HLA's main drawback is that it can be very difficult to modify existing simulations to conform to its specification. Many fields make heavy use of COTS software and do not have the luxury of source code availability. Additionally, HLA can be an inefficient means of combining federates together. The system is based on a publish-subscribe mechanism where any federate subscribing to another will receive all of its updated information whether it is needed or not.

With this in mind, we created a federated simulation system that operates in the spirit of the HLA, but that uses our own interface for easier implementation in the current EPOCHS implementation. EPOCHS' architecture is shown in Fig. 1. This system consists of the following components:

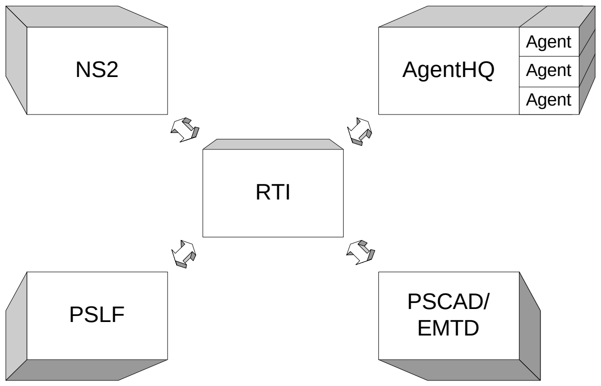

Fig. 1 The Relationship Between EPOCHS's Five Components

- **PSCAD/EMTDC**

PSCAD/EMTDC is used for electromagnetic transient simulation. EMTDC is a well-known electric power simulator. One of its main strengths is its ability to accurately simulate power system electromagnetic transients. That is, EMTDC models short-duration time-domain electric power responses. PSCAD is a graphical interface that is used to simplify the development of EMTDC scenarios. PSCAD is produced by the Manitoba HVDC Research Centre (Manitoba HVDC Research Centre, 1998).

EMTDC simulates power system scenarios in continuous time by solving a series of differential equations in a time-stepped manner. It has very detailed electrical models making it well-suited to electromagnetic transient investigations.

- **PSLF**

PSLF is used for electromechanical transient simulation. PSLF can simulate power systems with tens of thousands of nodes and is widely used by electric utilities to model electromechanical stability scenarios (General Electric, 2003). It models large systems in less detail than that available in PSCAD/EMTDC making it better-suited for long-running scenarios. It simulates power systems in continuous time by solving differential equations in a time-stepped manner that is similar to that employed by PSCAD/EMTDC.

- **NS2**

Network Simulator 2 (NS2) is an event-driven communication network simulator that was created through a joint effort between the University of California at Berkeley, Lawrence Berkeley Labs, the University of Southern California, and Xerox PARC. NS2 is a high-quality simulator that allows the creation of a wide variety of communications scenarios. It has built-in support for the most widely used network protocols and for the most popular research protocols. It is particularly valuable when studying TCP/IP behavior within IP-based networks due to its detailed models (Breslau, 2000). NS2 is able to simulate the behavior of these protocols even under various forms of stress, such as might be caused by competition for network resources when multiple applications share a network and communicate over the same routers and communication links, the impact of failures including router failures, link failures, or denial of service attacks. It can even capture the normal dynamics resulting from relaying messages with real-time data rates through many layers of routers. This is particularly important due to TCP's early adoption in new standards such as the Utility Communications Architecture (UCA) within the electric power community (Adamiak, 1999).

- **AgentHQ**

AgentHQ is a module that we developed to present a unified environment to our agents and acts as a proxy for those agents when interacting with other EPOCHS components. Through it, the agents can get and set power system values and send and receive messages to one another. AgentHQ is a discrete-event system. Events are processed as they occur and routed to the agents that are affected by them.

- **Runtime Infrastructure (RTI)**

The RTI acts as the "glue" between all other components. It is responsible for simulation synchronization and for routing communication between EPOCHS components. A firm requirement placed on any simulation system is that no event can be processed with a time stamp earlier than one that has already been completed. This makes good sense and is easy to enforce in sequential simulators, but issues arise when making use of distributed simulation systems. Many methods exist for dealing with this matter in the parallel and distributed simulation research community (Fujimoto, 2000). We employ a time-stepped model, one the simplest techniques for component synchronization, in our current system when running EPOCHS scenarios. Time steps are user-selectable and can be chosen depending on the granularity of a given case. Simulations can use a short time between synchronization points to compensate for the errors introduced by the decoupled simulation approach or can use larger time steps for faster execution.

2.4 Component Interaction

The synchronization between the various simulation components follows a simple algorithm. All systems are halted at time 0. At the beginning of any time step, the RTI waits for synchronization messages from both the power system simulator and NS2. Then, the RTI yields control to the AgentHQ. The AgentHQ passes the control on to the agents one by one until all have had a chance to execute. During this cycle, the agents are capable of sending communication messages and getting/setting power system variables. Once all agents are done, the AgentHQ returns control back to the RTI. Finally, the RTI notifies both NS2 and the power system simulator that the current time step is done. At this point, the two simulation engines run for an additional time step. Special attention must be paid to NS2. Messages may be received in between two synchronization points within NS2. If a message arrives, NS2 will immediately pass it along to the RTI bound for the AgentHQ. The AgentHQ will, in turn, pass the message on to the appropriate agent. The agent can process the message and send another in response. If the message requires power system state to be read or changed then that agent keeps the message in a queue until the next synchronization point occurs.

2.5 Simulation Scripts

Each agent simulation takes three parts. The structure of the power system and its electrical parts must be laid out in a PSCAD/EMTDC or PSLF compatible file. The layout of the communications subsystem and the transport protocols used needs to be specified in an NS2 file. Finally, agent types and locations are added to the NS2 simulation script for use by the agent manager. If the agent component were decoupled from NS2 then this would result in a third simulation file that would be required by EPOCHS. This is one limitation inherent in combining multiple simulators together. Each must be initialized in its own way. The other major drawback is the extra communication and management overhead between the simulation components.

2.6 Implementation and Optimization

In the current implementation, NS2, the RTI, the AgentHQ, and its corresponding Agents are all combined inside a single executable. Each component is logically separated within the source code and the RTI is still implemented as a protocol stub inside NS2. This combination was used purely to boost the performance of the simulation federation.

We have used EPOCHS to investigate a number of power system situations. None of them use PSLF and PSCAD/EMTDC at the same time. It is technically possible to do so, however we have not encountered a good example where this would be beneficial. As a result, we did not allow both PSCAD and PSLF to operate as federates simultaneously, but it would only require a minor modification to do so.

3 FEDERATING COTS COMPONENTS

PSLF and PSCAD/EMTDC are COTS programs and their source code is not available to the general public. NS2 is a research system, and its source code is available. However, it would require a great deal of effort to understand its more than 150,000 lines of source code sufficiently well to modify it to interface with an RTI. As an alternative, we used internal API's to federate each of these components. Strabburger listed four standard methods for making a simulation compliant with an RTI approach to simulation federation in (Strabburger, 1999).

They are, from most to least desirable:

- Re-implementation of the tool with the proper extensions
- Extending the simulation with intermediate code
- Using an external programming interface
- Coupling via a gateway program

In the first approach, simulation developers modify a simulator's internal source code so that it can interface with the RTI. In the second approach, if the simulator in question generates intermediate source code in a higher level language then the developer can include source modules of her own design to add RTI support. Some tools include the option of calling arbitrary functions either in user-specified source code or in dynamic link libraries. The third option takes advantage of this facility to add RTI support. Finally, if none of the previous options are available, but the simulation engine in question includes facilities for external communication via files, pipes, network communication, or some similar means then the developer can use that method to communicate with an external gateway program that will process commands and pass their results along to the RTI. The federation of the three commercial and research systems serve as an interesting case study because each of them used a different technique from Strabbugers' list.

Our synchronization approach allows us to use a convenient alternative to modifying the core source code. PSCAD/EMTDC, PSLF, and NS2 allow user-defined extensions. That is, a PSCAD/EMTDC scenario can include user-defined libraries that add equipment definitions using the C programming language that were not present in the original software. PSLF similarly allows user-defined equipment models using its proprietary interpreted EPCL language. NS2 has well-defined procedures for adding new communication protocols in C++ to the base simulation software. We have created our own equipment stubs whose sole purpose is to interact with EPOCHS's RTI at each synchronization point. Both PSCAD/EMTDC and PSLF use a user-modifiable length between each of their time steps. Both systems allow users to modify the time step length at each interaction, however we chose to keep time steps consistent for easy interaction in our first EPOCHS release.

This process is simplified by the fact that PSCAD/EMTDC, PSLF, and NS2 are all single-threaded systems, so each system is effectively halted whenever a synchronization event takes place. Additional efforts would be required if that were not the case.

The federation techniques used were:

- **NS2**

The network communication component uses an approximation of the second approach of integrating COTS, or more accurately research system, components. A new transport protocol is added to NS2 serving as its link to the RTI. A periodic call is added to the simulation script invoking the new protocol in order to halt execution and interact with the RTI once per time step. The length of the step can take on any value as long as it is the same as that used in the power system simulator. NS2 normally lacks the ability to automatically track message contents under most circumstances when using TCP/IP. We use the TCPApp application in order to keep track of this state on our behalf. UDP, by contrast, does have the ability to transmit data and we took advantage of it adding our own layer of abstraction through a module we named UDPApp. These choices gives us a great deal of flexibility as we can easily select any communication protocol at any time by sending data through simple NS2 function calls.

- **PSCAD/EMTDC**

PSCAD/EMTDC uses the third COTS federation method. PSCAD/EMTDC generates FORTRAN source code based on scenarios created in its graphical environment. Users can extend its functionality by making calls to source code written either in the C or FORTRAN languages and this code is compiled in with the generated code. Calls to this extended source code can be embedded into PSCAD scenarios, but unfortunately the stub must be customized to each scenario since the stub must access each internal variable by name. PSCAD/EMTDC is a continuous-time system. We use a library in our simulations that adds a call to our user-defined component once per time step. The PSCAD/EMTDC component begins by reading in all user-accessible equipment values that might be requested. Next, the electrical component contacts the RTI and notifies it that the beginning of the time step has been reached. Agents can request equipment values or can set power values when they execute. At the end of an agent execution cycle, a `finish` message is sent from the RTI to the electrical components and the power component set any values that have changed in their simulations. The components relinquish control afterwards and execution continues.

- **PSLF**

In PSLF, we use an approximation to the fourth COTS federation method. PSLF includes its own native language called EPCL that is roughly similar to C. Using this language, we were able to create our stub enabling us to interact with the RTI using files. The use of files was necessary because no other method was supported by the language. It would have been possible to go through a gateway program to translate these files into TCP/IP transmissions if we wished to communicate with modules in other systems, but we chose to run all simulations on the same machine making this step unnecessary.

PSLF halts execution at periodic intervals and waits for requests from the RTI. Incoming requests to get or set electrical simulation values are processed and the results are returned to the RTI. When all requests have been fulfilled, a final message is sent to PSLF allowing it to continue execution.

4 AGENT FRAMEWORK

4.1 Agent Definition

The keyword 'agent' has been popularized both in and out of the artificial intelligence research community, but there is still no single universally accepted definition about what it means to be an agent. Agents nearly always have the properties of autonomy (the ability to take independent action) and interaction (the capacity to sense the surrounding environment and make changes to it). In addition, an agent may exhibit the properties of mobility, intelligence, adaptivity, and communication. In this paper, the term agent will be used to refer to computer programs that are autonomous, interactive, and have the ability to communicate over a network. Agents may optionally also have any of the other attributes defined above.

4.2 Related Work

A wide range of simulations have made use of agents. There are two main classes of agent-based simulators. The first class uses agents to act as a mechanism for combining simulation engines. The flexibility that agents provide can be used to more efficiently link simulation components together such as by using filters to reduce inter-simulation traffic. An example of this can be found in (Wilson, 2000). The second class of simulations use agents to model entities within a simulated world. (Lee, 2001) makes use of a mix of continuous and discrete-time simulators in an air-traffic control simulation using object-oriented agents to represent, among other things, air-traffic controllers, aircraft, and air traffic generators. Our work is similar in spirit to that done by Lee, but it makes use of commercial off-the-shelf and research systems and is targeted towards electric power systems making use of network communication. These are just two of many examples of both classes of agent-based simulations.

4.3 Agents in Electric Power Protection and Control Systems

The electric power grid has traditionally been made up of a large number of protection and control devices that act on local information to respond to problems. This method works well in some cases, but is inefficient in many others. Agents have begun to be recognized as a natural solution to this problem in the electric power research community. Their autonomous nature, ability to share information and coordinate actions, and the potential to be easily replaced from remote facilities make them potentially valuable.

The protection and control scenarios that interest us use geographically distributed agents located in a number of Intelligent Electronic Devices (IEDs) as shown in Fig. 2. An IED is a hardware environment that has the necessary computational, communication, and other I/O capabilities needed to support a software agent. An IED can be loaded with agents that can perform control and/or protection functions. These agent-based IEDs work in an autonomous manner where they interact both with their environment and with each other. An example of this might be digital relays where each one has its own thread of local control, but they perceive a more global scope of the system and act in response to their non-local environment by communicating with other agents either via Local Area Networks (LANs) or via Wide Area Networks (WANs). These IEDs are relatively rare at present, but many are already available and we expect their use to increase over time.

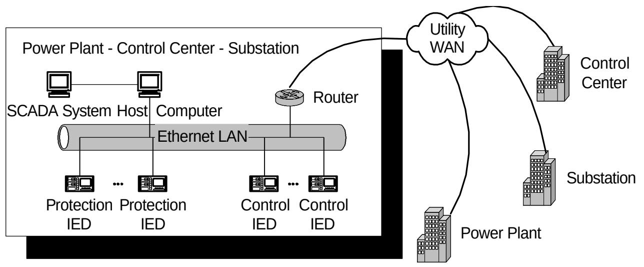

Fig. 2 Placements of the Agent-based IEDs within the Utility Intranet Infrastructure

The agent-based IED's structure is depicted in Fig. 3. Agents within an IED perceive their environment through local sensors and act upon it through the IED's actuators. Examples of sensor inputs might include local measurements of the current, voltage, and breaker status. Actuator outputs might include breaker trip signals, adjusting transformer tap settings, and switching signals in capacitor banks. Agents might even interface with legacy systems such as Supervisory Control and Data Acquisition

(SCADA) systems. The host computer shown in Fig. 2 could act as a bridge between the old and new systems in this type of situation. Internally, agents might be composed of many layers of functionality and control or may be contained in a single layer depending on the designer's specifications and implementation. As shown in Fig. 2, agents have the ability to communicate through a LAN in order to interact with other agents directly located on that same LAN, or can pass information along to the Utility WAN, i.e. the Utility Intranet, ultimately communicating with more remote IEDs.

The rising use of agents in IEDs makes an agent-based framework a natural choice. Protection and control engineers can create agents for use in real situations and test them with minor modification in the EPOCHS environment. Of course, agents can also mimic the behavior of more traditional systems. EPOCHS's early adopters have found the agent concept to be an intuitive one.

4.4 The Structure of a Utility Communication Network

Networked computing systems are becoming increasingly prevalent in many areas and we believe that this growth will occur within electric utility systems as well. Technology is constantly changing, but we can make some educated guesses about what utility communication systems will look like. First, the network systems will almost certainly be built from standard commercial off-the-shelf components. To do otherwise would be expensive both in terms of initial cost outlay and system maintenance. This means that these networks will be based on Internet standards even if the systems remained independent of the global network conglomeration. We can already see hints that such changes are coming in recent standardization efforts such as the Utility Communications Architecture (UCA). We believe that IP-based communication protocols will be heavily used in utility communication for these reasons.

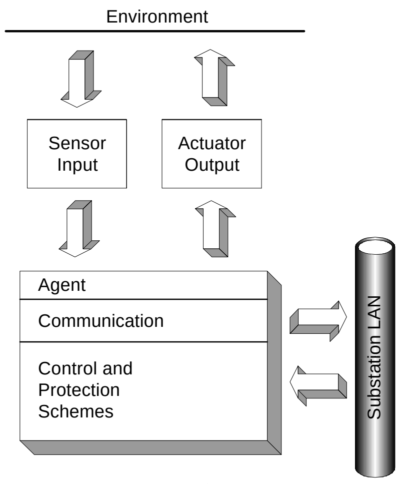

Fig. 3 The Structure of an Agent-based IED

4.5 Agent-based Simulation Framework

We have created a basic simulation framework both to minimize the changes needed between simulated systems and their real-world counterparts as well as to ease implementation for EPOCHS' users. This framework provides basic functionality for the EPOCHS agents. The basic functionality is shown in Fig. 4. Functions have been broken down into two main categories. Events occur in the AgentHQ subsystem and notification is passed on to the appropriate agents. Agents interact with their surroundings by using straightforward method calls both in order to get and set the state of their environment and to exchange messages between themselves.

AgentHQ is triggered at each synchronization point and acts as a proxy between the agents, the network communications simulation, and the electric power simulators. At that time, the AgentHQ calls each of the agent's `action` methods giving them an opportunity to calculate their set of operations for the next time step.

```
Interface Agent
{
methods:
    double get_round_time();
    void send_comm._msg(comm._type, group,
                         src, dst, pkt_size,
                         msg);
    void send_power_msg();
    void recv_power_msg();
events:
    void request();
    void action();
    void recv_comm._msg(comm._type, group,
                         src, dst, pkt_size,
                         send_time, round_num,
                         msg);
    void recv_power_msg(msg);
};
```

Figure 4: The Agent Interface

The agents remain dormant until they receive an event notification. Power system agents mimic those in real protection and control systems polling the current environment at regular intervals. When the beginning of an interval is reached, each agent is given a chance to `request` its power system state information through the use of the `send_power_msg` method and receive the results through the `recv_power_msg` event notification. All agents' initial requests are sent and the replies are received in one block. This is an optimization to help compensate for the use of files to exchange information between the AgentHQ and the electric power simulation systems. When all agents have been given a chance to run, the AgentHQ will give each of them a chance to react to their current state in the `action` method where they can send communication messages using the `send_comm_msg` method, or can make additional power system get and set requests using the `send_power_msg` method. In addition to these regular activation intervals, individual agents may receive communication messages through the `recv_comm_msg` event at any time and can take additional action in response to that event.

5 CASE STUDY – A SPECIAL PROTECTION SYSTEM

Power system generators are run synchronously. When one or more generators lose synchrony, the resulting transient instability can lead to costly blackouts. Stability problems are often caused by disturbances such as the loss of generation, loads, or tie lines. These disturbances stimulate power system electromechanical dynamics, resulting in deviations in frequencies, voltages, and generator phase angles. Special Protection Schemes (SPS) are devices that are most commonly designed to counteract instances of power system instability. Most SPS schemes do so by using a combination of generation rejection and load shedding (Anderson, 1996).

Traditional SPS systems are based either on purely local measurements to detect transiently unstable situations, a source of unreliability, or on communication that is too slow to allow them to respond to many types of faults. We created an agent-based SPS system that used wide-area measurements in a novel frequency prediction and control algorithm. Results showed that the system was successfully able to keep a system transiently stable and maintain its frequency above a preset threshold through rapid generation rejection. The precise load shedding required was calculated and acted upon in a single step. This would not be possible without the use of wide-area measurements. These experimental results showed the accuracy and usefulness of the method when communication channels were lightly loaded. However, inaccuracies were introduced under heavier communication traffic levels. Frequency levels decline as time passes after a fault making it important that measurements are used close to the time that a fault occurs. Under heavy traffic conditions, it may not be possible to detect when the fault occurred due to a lack of data points making it more likely that a stay value is introduced. This outcome points towards issues that must be dealt with in future research before protection and control systems utilizing Internet technology are ready to be deployed.

By using the agent-based framework provided, both traditional and agent-based systems were able to be created in an environment that isolated the modeler from the details involved in coordinating the various COTS and research components. EPOCHS's users could work in the framework provided with few reminders that these components were distributed making it much easier to concentrate on their work. The authors believe that this resulted in fewer bugs and faster modeling time than would have been otherwise possible. The use of EPOCHS allowed designers to understand the issues involved in creating power protection and control systems that had to take inherent delay, unpredictable message delivery times, and the possibility of message loss due to congested network conditions into account. This was an eye-opening experience in some cases. The experimental results received would have been difficult to reproduce using other tools and helped validate the concepts behind the EPOCHS project.

The complete set of results for the special protection system is outlined in greater detail within the chapter entitled, "Agent Technology Applied to the Protection of Power Systems" in (Thorp, To Appear in 2003). Two additional protection systems and their experimental results running on the EPOCHS platform are also included.

6 CONCLUSION

In this paper we have described EPOCHS, a simulation engine that combines PSCAD/EMTDC, PSLF, and NS2 functionalities together with an agent component. The paper has three main contributions. First, the simulator is the first to combine realistic network communications with electric power components. Second, EPOCHS serves as a case study illustrating methods for bridging unrelated simulation engines without making use of source-code modification to any of the systems in question. Finally, we make use of a simple yet powerful agent framework that is easy to use for modelers making use of EPOCHS.

We feel that techniques like those used in EPOCHS will become more common over time as commercial/open-source software continues to improve in terms of cost, availability, and feature set.

ACKNOWLEDGEMENT

The authors were supported, in part, by DARPA under AFRL grant RADC F30602-99-1-0532 and by AFOSR under MURI grant F49620-02-1-0233. They were also supported by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo, Brazil) and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brazil).

7 REFERENCES

Adamiak M., W. Premeriani 1999. Data Communications in a Deregulated Environment. IEEE Computer Applications in Power 12 (3): 36-39.

Anderson P., B. K. LeReverend 1996. Industry Experience with Special Protection Schemes. IEEE Transactions on Power Systems 11 (3): 1166-1179.

Breslau L., D. Estrin, K. Fall, S. Floyd, J. Heidermann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, H. Yu 2000. Advances in Network Simulation. IEEE Computer 33 (5): 59-67.

Fujimoto R. M. 2000. Parallel and Distributed Simulation Systems. New York, NY: Wiley-Interscience.

General Electric 2003. PSLF Manual. Available online via <http://www.gepower.com/dhtml/corporate/en_us/assets/software_solns/prod/pslf.jsp> [accessed March 12, 2003].

Kuhl F., R. Weatherly, J. Dahmann 1999. Creating Computer Simulation Systems: An Introduction to the High Level Architecture. Upper Saddle River, NJ: Prentice Hall.

Lee S., A. Pritchett, D. Goldman 2001. Hybrid Agent-based Simulation for Analyzing the National Airspace System. In Proceedings of the Winter Simulation Conference, Arlington, VA, USA, 1029-1037.

Manitoba HVDC Research Centre 1998. PSCAD/EMTDC Manual Getting Started. Winnipeg, Manitoba, Canada.

Strabburger S. 1999. On the HLA-based Coupling of Simulation Tools. In European Simulation Multiconference, 45-51.

Taylor S. J. E., R. Sudra, T. Janahan, G. Tan, J. Ladbrook 2001. Towards COTS Distributed Simulation Using GRIDS. In Proceedings of the Winter Simulation Conference, Arlington, VA, 1372-1379.

Thorp J. S., X. Wang, K. M. Hopkinson, D. Coury, R. Giovanini, To Appear in 2003. Agent Technology Applied to the Protection of Power Systems in Autonomous Systems and Intelligent Agents in Power System Control and Operation, Editor Christian Rehtanz, Baden, Schweiz: Springer-Verlag.

Tucci M., R. Revetria 2001. Different Approaches in Making Simulation Languages Compliant with HLA Specifications. In Proceedings of the Summer Computer Simulation Conference, 622-628.

Wilson L. F., D. Burroughs, J. Sucharitaves, A. Kumar 2000. An Agent-based Framework for Linking Distributed Simulations. In Proceedings of the Winter Simulation Conference, Orlando, FL, USA, 1713-1721.

8 AUTHOR BIOGRAPHIES

KENNETH M. HOPKINSON received his BS degree in Computer Science from Rensselaer Polytechnic Institute in 1997. Since that time, Ken has been working towards his Ph.D. degree in Computer Science at Cornell University. His areas of interest include distributed computer systems, simulation, and network communications protocols. Ken is currently investigating communications issues arising in the power grid under deregulation. He can be reached by e-mail at <hopkik@cs.cornell.edu>.

RENAN GIOVANINI received a B.Sc. degree in Electrical Engineering and M.Sc. degree from the EESC - University of São Paulo, Brazil in 1998 and 2000, respectively. He is presently a Ph.D. student at EESC - University of São Paulo, Brazil. His main research interests are power systems protection and applications of Artificial Intelligence. He can be reached by e-mail at <renan@sc.usp.br>.

XIAORU WANG received B.E. and M.E degrees from Chongqing University, China, in July 1983 and July 1988 respectively, and a Ph.D. from Southwest Jiaotong University (SWJTU), China, in June 1998 in Electrical Engineering. She is a Professor at SWJTU and is a visiting Professor at the School of Electrical Engineering at Cornell University. Her areas of interest include power system protection and substation automation systems with a focus on the application of wavelets and agent technology. She can be reached by e-mail at <xw44@cornell.edu>.

KENNETH P. BIRMAN is a Professor of Computer Science at Cornell University. A Fellow of the ACM, Birman has published extensively on reliable, secure distributed computing since joining Cornell in 1982. He developed the Isis Toolkit, which controls communication in the New York Stock and Swiss Exchanges, the French air traffic control system, and oversaw development of Cornell's Horus, Ensemble and Spinglass systems. He was Editor in Chief of ACM Transactions on Computer Systems from 1994-1999. He has advised the government on hardening the nationally critical communications infrastructure, and was co-chair of the 1995 DARPA ISAT study that set the research agenda in this field. He can be reached by e-mail at <ken@cs.cornell.edu>.

JAMES S. THORP is the Charles N. Mellowes Professor in Engineering and Director of the School of Electrical Engineering at Cornell University. In 1976, he was a faculty intern at the AEP Service Corporation. He was an associate editor for IEEE Transactions on Circuits and Systems from 1985 to 1987. In 1988, he was an overseas fellow at Churchill College, Cambridge, England. He is a member of the National Academy of Engineering, a Fellow of IEEE and a member of the IEEE Power System Relaying Committee, CIGRE, Eta Kappa Nu, Tau Beta Pi and Sigma Xi. He can be reached by e-mail at <jst6@cornell.edu>.

DENIS V. COURY was born in Brazil, in 1960. He received a B.Sc. degree in Electrical Engineering from the Federal University of Uberlandia, Brazil in 1983, a MSc degree from the University of São Paulo, Brazil in 1986 and a Ph.D. degree from Bath University, England in 1992. He joined the Department of Electrical Engineering, University of São Paulo, São Carlos, Brazil in 1986, where he is an Associate Professor in the Power Systems Group. He spent his Sabbatical at Cornell University in 2000. His areas of research interest are Power System Protection as well as new techniques for Power System Control and Protection including the use of Expert Systems and Artificial Neural Networks. He can be reached by e-mail at <coury@sel.eesc.sc.usp.br>.