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

EPOCHS: INTEGRATED COMMERCIAL OFF-THE-SHELF SOFTWARE FOR AGENT-BASED ELECTRIC POWER AND COMMUNICATION SIMULATION

Kenneth M. Hopkinson
Kenneth P. Birman

Computer Science Department
Cornell University
Ithaca, NY 14853, U.S.A.

Renan Giovanini
Denis V. Coury

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

Xiaoru Wang
James S. Thorp

School of Electrical Engineering
Cornell University
Ithaca, NY 14853, U.S.A.

ABSTRACT

This paper reports on the development of the **E**lectric **Po**wer and **C**ommunication Sync**h**ronizing **S**imulator (EPOCHS), a distributed simulation environment. Existing electric power simulation tools accurately model power systems of the past, which were controlled as large regional power pools without significant communication elements. However, as power systems increasingly turn to protection and control systems that make use of computer networks, these simulators are less and less capable of predicting the likely behavior of the resulting power grids. Similarly, the tools used to evaluate new communication protocols and systems have been developed without 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 the **E**lectric **Po**wer and **C**ommunication Sync**h**ronizing **S**imulator (EPOCHS), 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 simulator is viewed as best within its class for some category of uses. For example, 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 particularly good simulations of standard protocols like TCP and of standard Internet topologies, such as transit-stub configurations. On the other hand, none of these simulation systems were designed for interoperability, posing a challenge that our work addresses.

We live in an increasingly interconnected world where multiple domains such as water, power, and network communication can each affect the other. Yet, few stand-alone simulators exist that capture these inter-domain worlds. Constructing a combined simulation engine is potentially time-consuming and expensive. This is particularly true when simulations have both continuous and discrete-event components. An alternative is to link multiple simulations into a distributed environment (federation). Using this approach to combine multiple simulators for use in inter-domain situations is becoming more common, and there are even proposals to standardize such architectures, notably the Department of Defense's (DOD's) High Level Architecture (Kuhl, et al. 1999). 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, source code is only rarely available, and

this stands as an obstacle to federation. In the scenario tackled by EPOCHS, source code was available for NS2 but not PSCAD or PSLF.

Prior research on federating COTS simulation software includes work on supply chain manufacturing and manufacturing simulation in the GRIDS project (Taylor, et al. 2001), (Tucci and Revetria 2001)'s HLA compliance project, and Strabburger's SLX federation (Strabburger 1999). In addition, NIST has developed the Distributed Manufacturing System (DMS) Adapter, a mechanism for distributed simulation similar to that provided by the HLA, but with a more manageable level of complexity that is targeted towards the manufacturing community (McLean and Riddick). In the following year, a supply chain simulation was created based on a combination of the HLA and the NIST DMS adapter making use of the Arena, ProModel, and VB Application commercial software systems (Venkateswaran, et al.). Despite these success stories, the documented use of commercial simulation systems in federations is still relatively rare. A common concern is that once a group of simulators has been federated, "casual" modelers may find the added complexity of the new simulation platform difficult to manage.

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 illustrates a style of federation 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 divided into six sections. In section 2, we review background material and present the system's architecture. The methods used to federate the commercial and high quality research simulation components are discussed in section 3. In section 4, we outline our agent framework. We go on to describe a case study that has been developed and run on the EPOCHS platform in section 5. The paper concludes in section 6.

2 EPOCHS

2.1 Motivation

The restructuring of the electric power system, the creation of competitive markets, and the introduction of new regulatory mechanisms are now well-established trends. Understanding how the restructured power grid will operate, and how to monitor and control it, are necessary preconditions to achieving reliability and fairness under the conditions that may arise in the field. Yet, electric power simulators today do not model the network communication patterns seen in modern protection and control systems.

Traditional protection systems base decisions on local measurements, and employ conventional control systems that operate over rather slow, predictable communication systems. 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 begun to operate close 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.

It is natural to assume that systems with more data, and operating over faster communications networks, will be more effective than their predecessors. Yet, the Internet was not designed for safety- and time-critical applications, and layering the needed mechanisms over Internet protocols such as TCP/IP is a non-trivial undertaking. New kinds of simulation and evaluation tools are needed that bridge the gap, providing high-quality simulations of electric power scenarios while simultaneously modeling the behavior of computer communications protocols in realistic networks confronted with realistic scenarios, including load surges, outages, and other forms of dynamic stress. Our work demonstrates that when this is done, potentially serious problems can be identified and resolved in the laboratory, avoiding potentially costly or damaging mishaps in the field.

2.2 Overview

EPOCHS links simulators using a Runtime Infrastructure (RTI) to allow modelers to investigate electric power scenarios that involve network communication. One goal of EPOCHS is to minimize the intrusiveness of the simulation platform for users unfamiliar with its components. EPOCHS does this 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, which hides the details in the other simulation components. In the case of electric power systems involving network communication, this agent framework is natural, since hardware support for software agents exists in the power system today.

From a modeler's perspective, EPOCHS seamlessly links its three off-the-shelf simulation systems, enabling them to investigate power protection and control scenarios which 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 is deployed in a power system, and is known to operate correctly provided that the input data used by the control algorithm is accurate. EPOCHS will let us understand how that protocol might behave over TCP if 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.

Although we have only just begun to use EPOCHS for these purposes, we are finding that the most obvious combinations of control policies with TCP could malfunction under conditions that are quite likely to occur in a power system network. This suggests the need for new control algorithms, for consideration of TCP variants that provide Quality of Service (QoS) properties guaranteeing sufficient bandwidth to critical applications, and may ultimately argue against the use of standard TCP-based communication protocols that lack QoS in protection and control systems.

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 used to combine individual simulations, known as federates, together into combined simulators known as federations. The "glue" holding 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 drawback is that it can be difficult to modify existing simulations to conform to its specification. Many fields make heavy use of off-the-shelf commercial software and do not have source code access. Additionally, HLA can be an inefficient means of combining federates. 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.

We created a federated simulation system that works in the spirit of the HLA, but that uses our own interface for easier implementation. EPOCHS' architecture is shown in Figure 1.

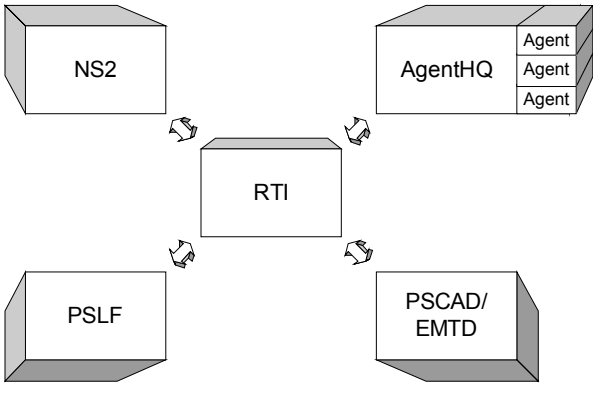

Figure 1: The Relationship Between EPOCHS's Five Components

Descriptions of the EPOCHS components follow.

PSCAD/EMTDC: PSCAD/EMTDC is used for electromagnetic transient **s**imulation. 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 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.

NS2: Network Simulator 2 (NS2) is an event-driven communication network simulator 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. Although there are many network and protocol simulators, NS2 is the most widely used simulator for evaluating the behavior of TCP, the TCP variants that have been proposed by researchers, and the behavior of routed UDP in large networks (Breslau, et al. 2000). NS2 is able to simulate the behavior of these protocols 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 also capture the normal dynamics resulting from relaying messages with real-time data rates through many layers of routers. We note that the proposed Utility Communications Architecture (UCA) is based on TCP (Adamiak and Premeriani 1999).

AgentHQ: AgentHQ is a module that presents a unified environment to agents and acts as a proxy when agents interact 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 affected agents.

RTI: The Runtime Infrastructure (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 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 have completed their tasks, 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 that power system state 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. Note that if the agent component were decoupled from NS2, a third simulation file would be required by EPOCHS. Script file generators are used to shield users from the details involved in defining these scripts to as large a degree as is practicable. However, this remains an inherent drawback in combining multiple federates together due to the inevitable differences between individual simulation models.

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 boosts the performance of the simulation federation.

Although we have used EPOCHS to investigate a number of power system situations, none uses PSLF and PSCAD/EMTDC simultaneously, and we have not encountered any problems for which this would be useful. Accordingly, the system is optimized under the assumption that only one power systems simulator is in use at a time.

3 FEDERATING COTS COMPONENTS

PSLF and PSCAD/EMTDC are commercial products and their source code is not available. NS2 is a research system, and its source code is available. However, it would require a great deal of effort to understand the 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:

- Reimplement the tool with the proper extensions
- Extend the simulation with intermediate code
- Use an external programming interface
- Couple 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 serves 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 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 effort would be required if that were not the case.

The federation techniques employed in EPOCHS are described below.

NS2: The network communication component uses an approximation of the second approach of component-level integration. A new transport protocol was added to NS2 to serve as its link to the RTI. A periodic call was 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 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 give us the flexibility to select any communication protocol at any time by sending data through NS2 function calls.

PSCAD/EMTDC: PSCAD/EMTDC uses the third 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 that interfaces with EPOCHS must be customized to each scenario since it 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 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 periodically 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

"Agents" are used extensively both in and out of the artificial intelligence research community, but there is no universally accepted definition for the term. 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 uses agents to act as a mechanism for combining simulation engines. The flexibility 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, et al. 2000). The second class of simulations use agents to model entities within a simulated world. (Lee, et al. 2001) employs 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 Lee's approach.

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 there are many situations in which information not readily available from local sensors or local databases would be needed to plan, to protect the grid, or to control it efficiently. Lacking such data, the grid must be operated in a very conservative and potentially inefficient manner, and might not be able to support desired behaviors. Agents have begun to be recognized as a natural way to introduce extensibility into the grid without drastically changing the usual power systems architecture, and are therefore gaining acceptance in the electric power research community. Their autonomous nature, ability to share information and coordinate actions, and the potential to be easily upgraded or controlled from a remote location are appealing to grid operators and protocol designers.

The protection and control scenarios that interest us use geographically distributed agents located in a number of Intelligent Electronic Devices (IEDs) as shown in Figure 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, interacting both with their environment and with one-other. For example, a digital relay could be implemented as an agent with its own thread of local control, but that also monitors conditions elsewhere in the network so as to act in response to their non-local events. This agent would communicate 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.

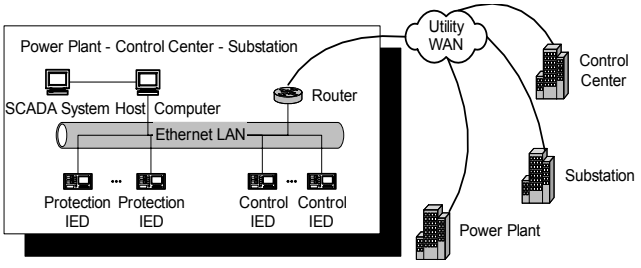

Figure 2: Placements of the Agent-Based IEDs within the Utility Intranet Infrastructure

The agent-based IED's structure is shown in Figure 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 current, voltage, and breaker status. Actuator outputs might include signals to initiate breaker trips, transformer tap setting adjustments, and capacitor bank selection. Agents might even interface with legacy systems such as Supervisory Control and Data Acquisition (SCADA) systems. The host computer shown in Figure 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 Figure 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.

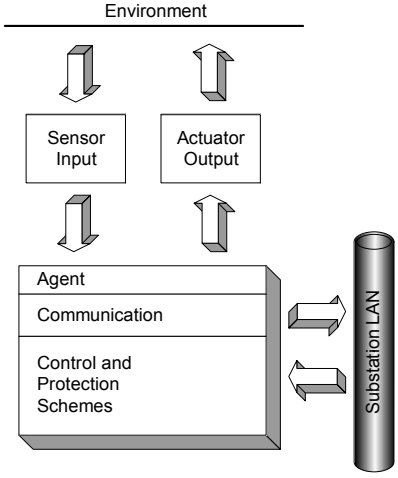

Figure 3: The Structure of an Agent-Based IED

The increasing 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 have become ubiquitous, and we believe that this trend will soon become more widespread within electric utility systems. 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 initial cost outlay and in 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 signs that such changes are coming in recent standardization efforts such as the TCP-based Utility Communications Architecture (UCA).

4.5 Agent-based Simulation Framework

Our framework is intended to minimize the differences needed between simulated systems and their real-world counterparts, as well as to ease implementation for EPOCHS' users. The basic functionality is shown in Figure 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 method calls to get and set the state of their environment and to exchange messages with other agents.

```
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

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 `request` and `action` methods giving them an opportunity to calculate their set of operations for the next time step.

The agents remain dormant until they receive an event notification. Power system agents mimic those in real protection and control systems by 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 allow each 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 actions in their response.

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 and LeReverend 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 results showed the accuracy and usefulness of the method when communication channels were lightly loaded.

Degraded SPS performance was observed when the communications network was subjected to increased communication traffic, as would surely arise in a setting where many kinds of applications share the utility communication infrastructure. Frequency levels decline as time passes after a fault, making it important that measurements from close to the time that a fault occurs are used in the SPS algorithm. Under heavy network traffic conditions, Internet routers are designed to drop data packets to signal overload to TCP endpoints. The protocols, sensing that congestion has occurred, will then slow down. These built-in design features of the network disrupt the SPS algorithm. Our findings suggest that the decision to base the UCA on TCP is potentially risky. In future work, we hope to evaluate

variants of TCP providing QoS guarantees (a great many have been proposed), while also exploring UDP-based communication protocols of our own design.

By using the EPOCHS agent-based framework, both traditional and agent-based systems were able to run in an environment that isolates the modeler from the details of coordinating the various COTS and research components. EPOCHS's users could work in our framework, with few reminders that these components were present under the surface. The authors believe that this simplified development, yielded a more robust solution, and reduced modeling time relative to other approaches. The use of EPOCHS allowed designers to focus on issues involved in creating power protection and control systems that take 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, et al. 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.

- The simulator is the first to combine realistic network communications with electric power components.
- 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.
- We make use of a simple yet powerful agent framework that is easy to use for simulation modelers.

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.

ACKNOWLEDGMENTS

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).

REFERENCES

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

Anderson P. M., 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*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith, 1029-1037, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

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

McLean C., F. Riddick. 2000. The IMS Mission Architecture for Distributed Manufacturing Simulation. In *Proceedings of the Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. R. Barton, 1539-1548, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

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*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith, 1372-1379, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Thorp J. S., X. Wang, K. M. Hopkinson, D. V. 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.

Venkateswaran J., M. Y. K. Jafferali, Y. Son. 2001. Distributed Simulation: An Enabling Technology for the Evaluation of Virtual Enterprises. In *Proceedings of the Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith, 856-862, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

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*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. R. Barton, 1713-1721, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

AUTHOR BIOGRAPHIES

KENNETH M. HOPKINSON received his B.S. degree in Computer Science from Rensselaer Polytechnic Institute in 1997. He went on to receive his M.S. degree from Cornell University in 2002. 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 electric power grid. <hopkik@cs.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 he oversaw development of Cornell's Horus, Ensemble and Spinglass systems. He was Editor in Chief of ACM Transactions on Computer Systems from 1994-1999. <ken@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 Artificial Intelligence. <renan@sc.usp.br>.

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 M.Sc. 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 Power Systems. His research interests center on Power System Protection and Control using techniques that include Expert Systems and Neural Networks. <coury@sel.eesc.sc.usp.br>.

XIAORU WANG received a Ph.D. in Electrical Engineering from Southwest Jiaotong University (SWJTU), China, in June 1998. 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. <xw44@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. 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. <jst6@cornell.edu>.