

# Extending EMTDC/PSCAD for Simulating Agent-Based Distributed Applications

Mesut Baran, Raghuram Sreenath, Nikhil R. Mahajan

Author Affiliations: Department of ECE, NC State University, Raleigh, NC, USA.

**Abstract:** This letter illustrates the development of an interface for the simulator EMTDC/PSCAD for integrating an agent-based distributed application into the simulation. The custom device modules the EMTDC supports have been used to provide a channel for the agents to communicate (send and receive data) with the simulator. The main advantage of this interface is that the agents now can be developed independently, as separate JAVA applications. The application used to illustrate the features of the interface is a pilot protection scheme over a line.

**Keywords:** EMTP, system simulation, protection.

**Introduction:** The advent of communication technology and the emerging agent-based software paradigm facilitate the development of new distributed protection and control schemes [1]-[3]. Development of distributed applications is, however, more challenging and simulation tools, such as EMTP, become essential. Current EMTP implementations do not facilitate the communication part very well, however. Since EMTP has been developed mainly for simulating the power system [4], it is implemented by using procedural languages such as Fortran or C. As a simulation environment, EMTP needs to be enhanced for protection and control schemes that use communication between devices.

This letter illustrates the development of an interface for EMTDC/PSCAD [5] to facilitate the simulation of agent-based distributed protection and control applications on the simulator. The interface can be applied to any simulator, which supports custom functions in C language.

**Simulator Interface for Agent Simulation:** To illustrate the interface needed to extend the EMTDC simulator, we considered a simplified pilot protection scheme over a line as the agent-based application [6]. Figure 1 shows the system to be simulated. Agents in this application are the protection relays, located at the two ends of the transmission line. To create a realistic simulation, the system and the agents will be simulated separately. The system can easily be simulated on the EMTDC. The relays will be implemented as separate software agents that can monitor the system and also send control (trip) signals to the system elements (the CBs in this case). Since, in actual implementation, the two relay agents will be communicating over a communication channel, the communication between the agents must be included in the simulation.

EMTP system simulation involves calculating the system response in time domain in small time steps. Therefore, we need an interface that will allow the agents to get the monitoring data from the system and pass the system their output (decision) at each time step. The EMTDC has such an interface, mainly for the development of “custom modules” by the users for simulating special devices. Figure 2 illustrates this interface. DYSDYN in the figure handles the user-defined subroutines. At each time step, the DYSDYN reads the network quantities (voltages and currents) it needs, processes the user-defined procedures, and gets the output of the user modules. EMTDC allows this interface to be implemented with either Fortran or C subroutines.

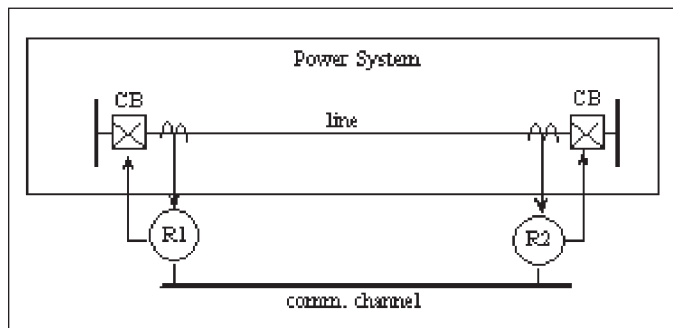


Figure 1. A transmission line with pilot protection

One approach to simulate the protection scheme would, therefore, be to write custom modules to represent the relays as custom devices, and thus “embed” the protection device simulation into EMTDC. But this approach would be quite restrictive, as the agents are usually developed using object-oriented languages like Java. Java also facilitates the communication between software agents by providing access to the network protocols that the operating system supports. Therefore, we used the custom modules for just providing a channel for the agents to communicate (to send and receive data) with EMTDC. The modules to handle the data exchange through DYSDYN are written in C language. This allows the data exchange to be through socket communication, since both C and Java support socket communication. This interface is illustrated in Figure 3. The main advantage of this interface is that the agents now can be developed independently as separate JAVA applications.

The modules put the data they want to send to their agents in a string, separated by commas. For the protection application considered, the format of the data is as follows:

<time, current, breaker\_state> e.g.: “0.115,1.5,0,”

which indicates that at time 0.115 sec the line current was 1.5 p.u., and the CB was closed. This message is sent through the socket communication channels, as Figure 3 illustrates. The message is processed on the agent side by a program called the server. The agent architecture developed for the pilot protection scheme is discussed next.

**Agent Architecture:** An agent distinguishes itself from any other distributed component in that it is capable of acting autonomously, i.e., decide the course of action that is in its best interest [7]. What this translates to in programming is that, instead of making method calls or initiating a particular procedure remotely, as is typical in a distributed computing environment, the communication is through messages sent to one another. These messages can be thought of as requests for actions, which may or may not be honored by the receiving agent. The decision is made by a policy defined for every agent.

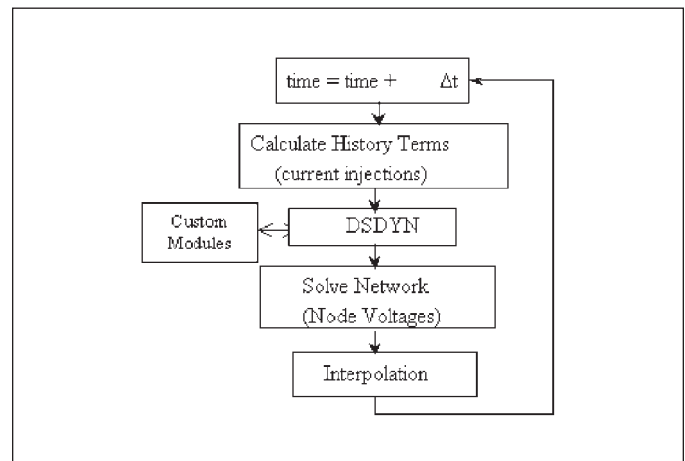


Figure 2. Main steps in EMTDC simulation

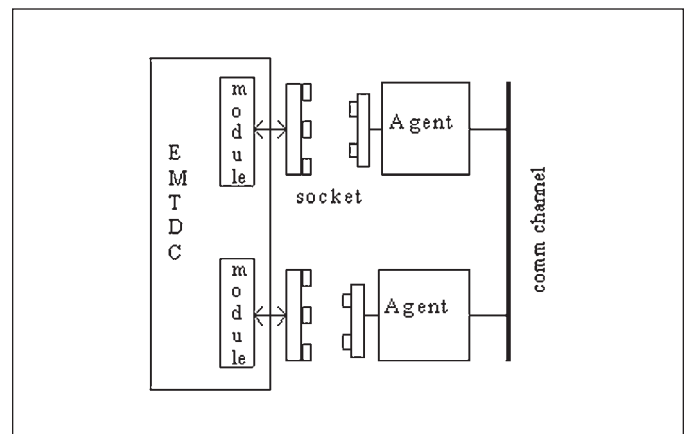


Figure 3. EMTDC—Agent interface through sockets

For the protection system, the agent architecture that we decided on using is typical of any “multi-agent system.” There is a backbone or a bus, which we call the “agent communication channel,” to which the agents can connect and send/receive messages to each other. These messages, whose structure is based on (but not exactly like) FIPA specifications [8], contains a “to address,” which is the agent I.D. (a universally unique I.D. for an agent) of the recipient agent. The actual delivery of the messages to the intended party is taken care of by the communication manager CM), thus freeing the sending agent from the burden of having to know the exact physical location of the receiving agent.

Figure 4 illustrates the basic structure of the relay agent. The CM and its associated queues together handle the agent communication.

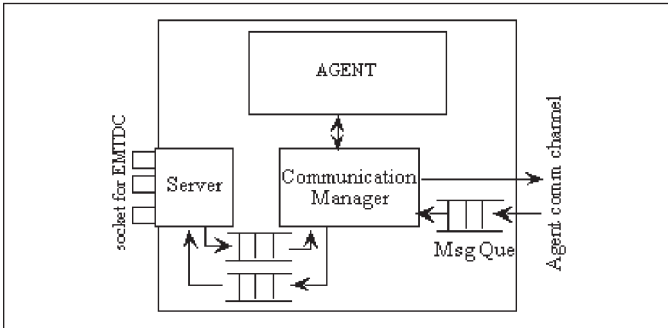


Figure 4. Agent and its communication components

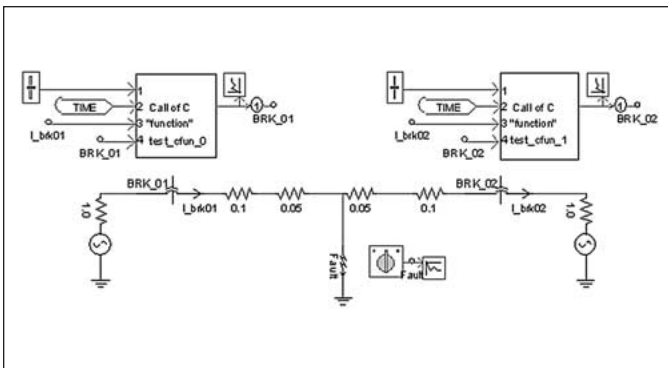


Figure 5. Pilot protection simulation on EMTDC

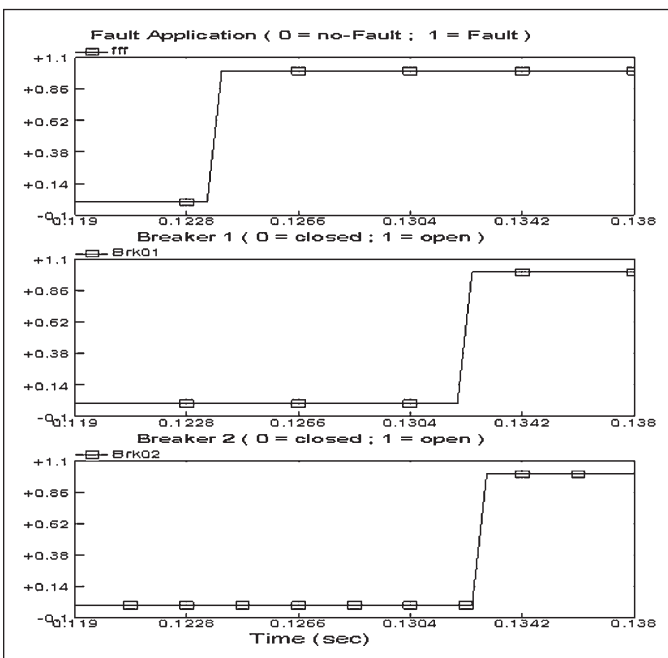


Figure 6. Fault and breaker profiles from EMTDC

The intercommunication between agents is through Java Remote Message Invocation (RMI), which is used as a means to populate the neighboring CM’s queue. For communication, we implemented a “token ring” type of protocol, in which each CM has only the knowledge of its neighbor, and any message not intended for the agent is passed to its neighbor.

The format of the messages between the agents is as follows:

<I.D., from, to, content, behavior, action>, e.g.:  
<546,Relay1,Relay2,“0.115,fault,”REQ-RES, REQ>

where message I.D. is a unique number for each message, content of the message contains the data the message carries, and behavior and action identifies the “message type,” i.e., it helps the receiving agent interpret the message. More examples are given in the following section to further illustrate how the agents communicate using these messages.

**Agent-Based Pilot Protection:** The pilot protection scheme, illustrated in Figure 1, consists of two relays placed at the two ends of a transmission line. These relays monitor the line and when they see a fault (high current) they communicate with each other in order to determine the location of the fault. Communication enables the relays to determine whether the fault is within their protection zone (i.e., on the line) or outside their protection zone. The communication between the relays initiated by one of the relays sending a message to the other asking whether it saw the fault. If so, then the fault is on the line, and thus both relays operate to isolate the fault. This simple communication improves the ability of the relays to locate the fault much more reliably than they could do otherwise [6].

Figure 5 shows the system simulated on EMTDC. The two blocks on top of the breakers (BRK-1, BRK-2) represent the custom interface modules that send and receive data from relay agents. As the figure illustrates, the agents get their current measurements from the system (EMTDC). When the current exceeds a threshold value, this triggers a protection action from the relay agents. The sequence of events that take place for fault clearing by the relay agents, and the communications between the agents to accomplish this task are as follows:

1. Agent relay1 notices that there is downstream fault and sends a request to the agent relay2 to see if it noticed the fault:

<312, R1, R2, “0.115,fault,” REQ-RES, REQ>.

The message type indicates that this is a request-response-type message and the R1 (relay1) is requesting a response from R2. The message content indicates that R1 saw the event “fault” at time 0.115 sec.

2. Agent relay2 replies saying that it saw the fault at 0.116 sec.:

<313, R2, R1, “0.116,fault,” REQ-RES, RES>.

R2, knowing that the fault is on the line, sends then a message to EMTDC to open the breaker BRK-2:

<314, R2, EMTDC, “OPN,BRK-2,” ACTION, REQ>.

This message is passed to the server by CM of R2, and the server in turn passes it to the EMTDC.

3. Agent R1 receives the message from R2 confirming that the fault is on the line and, therefore, it also sends a message to EMTDC to open the breaker BRK-1 it controls:

<315, R1, EMTDC, “OPN,BRK-1,” ACTION, REQ>.

Figure 6 illustrates the simulation results and confirms that the above sequence of events took place: first the fault is initiated (top figure), and shortly after the fault, the two CBs open (bottom two figures). The time delays between the fault and the CB openings reflect mostly the communication time delays. The simulations indicate that the interagent communication is quite fast, therefore intentional time delays are introduced to reflect more realistic delays.

The interface between the EMTDC and the agents is, however, quite slow relative to the agent communication. This is mainly due to the fact that the socket channel the data has to go through between the EMTDC and the agents. At the end of each time step, EMTDC opens the socket and sends the data, and gets the data from the agents. This data transfer through the sockets is quite slow; in our simulations it is about 0.35 sec. for each data transfer. This slow data interface, therefore, slows the EMTDC simulation time considerably, especially if the time steps are small.

**Conclusions:** An interface has been developed for EMTDC that allows the simulator to exchange data with programs written in JAVA. JAVA as an object-oriented language facilitates the development of agent-based applications. A basic agent-based relay protection scheme

illustrates that the new interface facilitates the simulation of multi agents with message communication features. This interface can be adapted to any EMTP type simulator which supports user-defined modules written in C language. The simulation results indicate that the interface is, however, slow due mainly to the socket communication the interface uses.

**Acknowledgments:** This work is sponsored by the office of Naval Research (ONR) under award no: N0000014-00-1-0475.

**References:**

[1] D.V. Coury, J.S. Thorp, K.M. Hopkinson, and K.P. Birman, "An agent-based current differential relay for use with a utility intranet," *IEEE Trans. Power Delivery*, pp. 47-53, Jan. 2002.  
 [2] S. Talukdar and E. Camponogara, "Network control as a distributed, dynamic game," *Proc. 34th Hawaii Int. Conf. System Sciences-2001*.  
 [3] R. Gustawson, "Agents with power," *Communications of the ACM*, pp. 41-47, Mar. 1999.  
 [4] J.A. Martinez-Velesco, *Computer Analysis of Electric Power Transients*. New York: IEEE Press, 1997.  
 [5] Manitoba HVDC Research Centre, "PSCAD/ EMTDC user's manual." Available: www.hvdc.ca  
 [6] S.H. Horowitz and A.G. Phadke, *Power System Relaying*. RSP Publishing, 2000, pp. 138-163.  
 [7] G. Weiss, *Multiagent Systems*. Cambridge, MA: MIT Press, 1999.  
 [8] Y. Labrou, et al., "Agent communication languages," *IEEE Intell. Syst.*, pp. 45-52, Mar./Apr. 1999.

**Copyright Statement:** ISSN 0282-1724/02/\$17.00 © 2002 IEEE. Manuscript received 5 July 2002. This paper is published herein in its entirety.

**PES Technical Committee Meetings**

*27-30 January 2003, Las Vegas, Nevada*

With the change from two to one General Meetings each year, PES realized there would be one less meeting each year for the Technical Committees that met at the PES General Meetings. PES decided that they would provide conference management and partial sponsorship one time for a meeting in January 2003 for all Technical Committees interested to meet in Las Vegas to further standards development. The Transmission and Distribution (T&D) Committee, Substations Committee, Energy Development and Power Generation (ED&PG) Hydroelectric Power Subcommittee, and Standards Coordinating Committee on Power Quality (SCC-22) will meet at the Riviera Hotel in Las Vegas, 27-30 January 2003. Detailed meeting schedules and registration information are available on the "Meetings" page of the PES Web site, <http://www.ieee.org/organizations/society/power/>. We encourage the other Technical Committees to participate. For more information, please contact:

- T&D Committee: Richard Piwko, +1 518 385 7610
- Substations Committee: John McDonald, +1 678 966 0363 x227
- ED&PG Committee: Steve Brockschink, +1 503 297 1631
- SCC-22: David Vannoy, +1 302 528 1938.

**Exact Pi-Model of UPFC-Inserted Transmission Lines in Power Flow Studies**

**Muwaffaq I. Alomoush**

**Author Affiliation:** Department of Electrical Power Engineering, Yarmouk University, Irbid, Jordan.

**Abstract:** The unified power flow controller (UPFC) combines advantageous features of both shunt and series compensations; therefore, it is an effective device that can adjust power system parameters in order to increase power transfer capability, stabilize the system, and improve the economics of a power system. This letter presents the exact pi-model of the UPFC-inserted transmission lines from which the injection-model using two-port networks is derived. The equations derived in this letter are applied to a small test system, where the impact of UPFC on OPF results is shown.

**Introduction:** With restructuring of the electric power industry [1], the transmission business undergoes an increased level of risk and uncertainty associated with transmission operations and investments that require more power flow control. Electric utilities are continuously looking for new devices that will enable interconnected systems to have increased power transfer capabilities with transmission lines. Traditionally, mitigating transmission constraints result in expensive or undesired alternatives, such as resorting to out-of-merit order on the generation side as a short-term solution or resorting to system expansion or up-rating on the transmission side as long-term solutions [2]. Increased attention should be given to new mechanisms to introduce more flexibility in the way the transmission system is operated and to new lower-cost mechanisms by which transmission constraints can be mitigated.

The UPFC is a FACTS device including shunt and series control elements and offers a unique combination of fast shunt and series compensations. It consists of two linked self-commutating converters with semiconductor devices with turn-off capability [2]-[4]. The converters share a common dc capacitor and are connected to the system through coupling transformers. The capacitor provides dc voltage support for converter operation and functions as an energy storage element. A UPFC injects a voltage in series with a transmission line through a series transformer. The active power involved in the series injection is taken from the transmission line through the shunt transformer, and the UPFC generates or absorbs the needed reactive power locally by the switching operation of its converters.

A UPFC offers flexible power system control with the advantage of providing, simultaneously and independently, real-time control of voltage, impedance, and phase angle, which are the basic power system parameters on which system performance depends [2]-[4], [6]. The UPFC can be utilized to control line active and reactive power, to achieve maximum power transfer capability, and to significantly improve power system reliability. In this letter we present the exact two-port model of the UPFC-inserted line that can be used directly in traditional power flow and OPF studies.

**Exact Two-Port Model of UPFC-Inserted Lines:** Figure 1 represents an equivalent circuit diagram of the steady-state model of a UPFC-inserted transmission line, where the UPFC is located between nodes *i* and *m* in the line, connecting buses *i* and *j*. This includes both UPFC and the pi-model of the transmission line. Each converter is represented by a voltage source in series with an impedance, where  $Z_{se}$  and  $Z_{sh}$  ( $Y_{se}$  and  $Y_{sh}$ ) are the impedances (admittances) associated with the

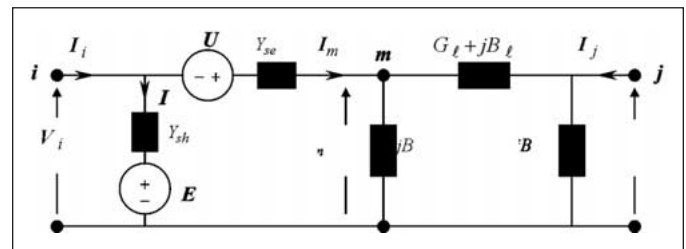


Figure 1. Representation of a UPFC-inserted line