Distributed Approaches for Determination of Reconfiguration Algorithm Termination

Pinak Tulpule, Karl Schoder, Ali Feliachi, Hong-Jian Lai
Advanced Power and Electricity Research Center, West Virginia University
Morgantown, WV 26506-6109

***Abstract*— The automatic reconfiguration of electric shipboard power systems is an important step toward improved fight-through and self-healing capabilities of all-electric naval warships. The authors have presented a distributed algorithm for autonomous reconfiguration within an energy management framework using an agent based approach. In the previous work, a blackboard architecture was used as globally shared memory structure for detection of algorithm termination. This paper presents alternative schemes for the detection of global properties, and specifically, detection of algorithm termination.**

***Keywords*—autonomous agent-based reconfiguration, distributed algorithms, shipboard electric power distribution systems, termination detection**

I. Introduction

The success of increasingly all electric naval warships requires new approaches to their design and operation. While the Naval Combat Survivability generation and propulsion and DC distribution testbeds [1] have proven that incorporating of the latest technology allows simplifying the design and maintenance of ships, the remaining control challenge requires advanced control strategies to ensure system stability and take advantage of the system's capabilities. The controls for the various power electronic devices and the automation of processes help system operators cope with routine operations, maintenance, and emergency situations. To optimize survivability lifetime, and cost performance, dependable automation strategies, nonlinear control theory, applied analytic redundancy, and distributed intelligence are envisioned to form robust networks and allow reconfiguration based on situational awareness. The realization of these goals is hindered by several technological shortcomings including incomplete theoretical basis and limited situational information. The challenge of providing a new and distributed answer for a decentralized energy management system (EMS) using a distributed and agent-based approach has been addressed in earlier work by the authors [2], and this contribution will extend the EMS concept by distributed algorithms that will provide alternative solutions for global termination detection.

Termination detection algorithms have a long history in the field of distributed computing and concurrent programming [3]. Termination is a global property and difficult to detect as each process has knowledge of only its local state and the states of processes to which it is connected through communication channels. References [3], [4], [5] give an extensive survey and comparison of various termination detection algorithms. One set of termination detection algorithms require a unique node to start computations and detect termination. Two examples for this type of algorithm, which is known as diffusing algorithm, are the Dijikstra-Scholten algorithm [4] and credit recovery algorithm [5]. Other algorithms are trying to take a consistent snapshot of the current computation to determine its status. Examples include the Snapshot algorithms [4] and Chandy-Lamport Algorithm [4]. Another class of algorithms allows any node to start the detection algorithm. Examples are wave based algorithms [5], and graph relabeling algorithm [6]. Other algorithms include the blackboard or shared memory approach used in earlier work by the authors [2].

The paper is organized as follows. Section II reviews the agent-based approach to the EMS and provides the problem statement. Section III gives an overview of different termination detection algorithms and Section IV explains the algorithm used in this work. Section V discusses some important aspects of the chosen algorithm and its implementation. Section VI presents an example for the reconfiguration of the shipboard power system. Conclusions are given in Section VII.

II. Problem Statement

A block diagram of the shipboard power system testbed is shown in Fig. 1. It includes two AC generators that supply power to the zonal DC distribution system through AC-DC converters (PS), and zonal loads that are represented through constant power loads (CPL) and induction motor drives (MC). Each generator also supplies power to the propulsion and pulsed loads. Most of the reconfiguration agents [2] are associated with one device, i.e., one per node as shown in Fig. 2, but the breaker agents have control of the two breakers next to their respective buses. Further details concerning an implementation of the shipboard power system prototype can be found in [7].

The main component of the energy management system is the distributed reconfiguration algorithm. The agents execute this algorithm acting locally in their environment to solve the power flow problem, i.e., supply load demands according to priorities. The agents are allowed to communicate and coordinate with their respective neighbors to reconfigure and find a globally acceptable solution. Therefore, the reconfiguration algorithm determines a solution to the power flow problem in a graph $G = (V, E)$ with vertices V (also referred to as

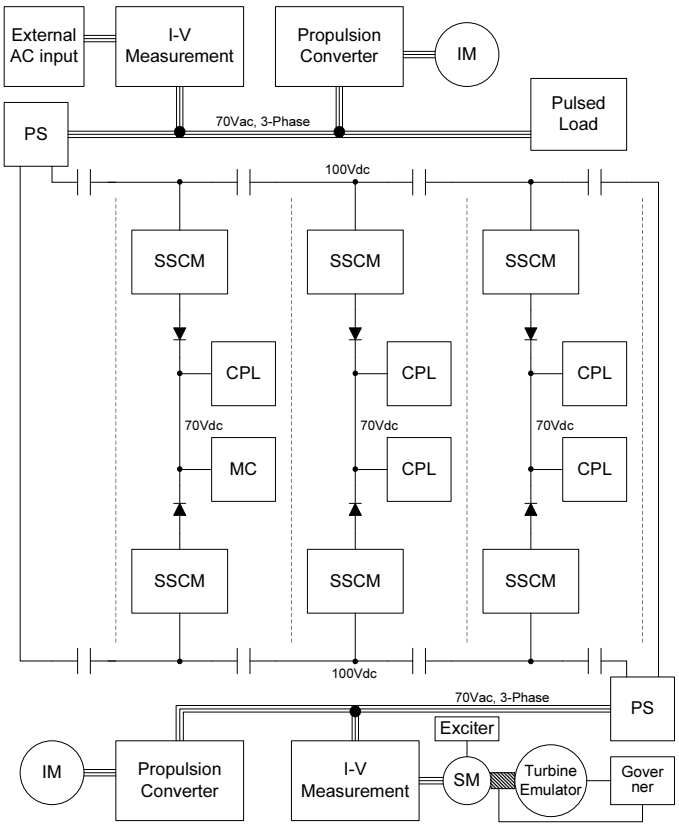

Fig. 1. Shipboard power system diagram

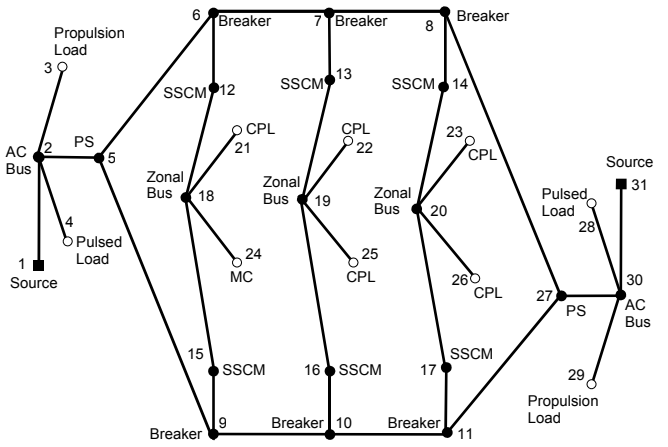

Fig. 2. Mapping devices to agents: ■ represents source nodes, • represents intermediate nodes and ○ represents load nodes

nodes, agents, and processes[1]) and edges E (also referred to as links and paths) that are electrically connected and have an associated communication channel. Each agent locally "knows" about its device, e.g., demand, demand priority, and status, and starts communicating with neighbors if it detects changes following the reconfiguration algorithm A. The algorithm A is based on a self-stabilizing maximum flow algorithm with nodes represented by software agents [8], [2]. The maximum flow algorithm consists of guarded statements to modify node and link variables such as flow, priority, residual capacity, supplied demand, and unsupplied demand. Each agent executes these guarded statements asynchronously and informs neighboring agents of any changes. An agent is idle if it does not have any statement to execute, and the algorithm converges when all agents are idle.

Even though the solution can be found through local actions, the implementation of the solution within the actual power system requires global data exchange using, for example, a blackboard to inform all the agents of convergence to a global solution. Only this globally shared memory allows a system-wide consistent transition from a previous power flow to a new power flow by, for example, changing switch positions, updating reference values, and sending commands to turn devices on or off. The problem addressed here is to find an alternative to the blackboard architecture. The key properties we are interested to improve are: first, avoid single-point of failure – a communication architecture with no backup blackboard would present such a single-point of failure problem, and second, do not just duplicate communication channels. Therefore, though duplicating communication hardware and infrastructure is feasible, simply keeping a "copy" of the current infrastructure and approach is not of interest here.

We start addressing the above specified problem by reviewing the literature on distributed systems and algorithms that already provide a rich body of alternative approaches and relate results to the shipboard power system reconfiguration agents.

[1]The terms nodes, agents, and processes come from the fields of graph theory, agent technology, and distributed algorithms, respectively.

III. Termination-Detection Algorithms

A. Dijikstra-Scholten Algorithm

Termination-detection algorithms for asynchronous networks with an underlying graph G with arbitrary undirected edges are available for diffusing algorithms A [4]. An algorithm A is said to be diffusing when all its activities start at a location where the input occurs and diffuses through some portion of the network via messages. A global state of A is said to be idle when no agent is performing any action and no message is in any communication channel between two agents. The termination-detection problem for A can be summarized as follows: if, sometime after an input occurs at some agent i, algorithm A ever reaches a quiescent global state, then eventually a special $done_i$ output should be performed at node i.

This Dijikstra-Scholten algorithm [4] constructs a spanning tree of the agents currently involved in A along which the messages are sent. Furthermore, each agent acknowledges messages in A immediately but the first. The sender of the first message is known as parent node. The tree is allowed to shrink again when two conditions are fulfilled at node i: first, agent i is idle, and second, all its outgoing messages have been acknowledged.

This type of termination-detection algorithm can be interleaved with our current reconfiguration algorithm at every node to form an extended algorithm $B(A)$. Due to the ability of each agent to detect changes in the electric power system or newly chosen set-points, every agent that is idle but detects a change in its local environment would have to trigger the termination-detection algorithm and report at its end to all the other agents that are part of the power flow solution. The set of agents that have to be informed does not necessarily need to include all agents but rather all the agents that form a self-consistent set of demand, routing, and supply agents. This feature would allow the system to "separate" into several consistent islands.

B. Snapshot Algorithms

Another algorithm type we are interested in belongs to the family of global snapshot algorithms [4]. A snapshot is "consistent" if it looks to the processes as if it were taken at the same instant everywhere in the system. Again, the original reconfiguration algorithm A can be augmented by the snapshot algorithm to form $B(A)$. The snapshot algorithms can be used for several purposes including determination of global properties, distributed debugging, and stable property detection. The stable property detection can be used for the special purpose of detecting termination. Two possible approaches are discussed next.

1) Logical-time Snapshot Algorithm: Due to the problems with real time in distributed systems, attempts have been made to create different concepts. One of them is the logical time by Lamport. The idea is to assign to every event of an executing algorithm a "logical time," for example, a number from the set of nonnegative integers. These assigned logical times need not relate to real time but need to keep the ordering among interdependent events. Lamport's algorithm maintains local clocks and advances them with every internal (executing a local action or sending a message) and external (receiving a message) event.

The agents originally following algorithm A are transformed to a Lamport-time process $LT(A)$ by adding a local clock variable that is initially set to zero and increased by one at every event that occurs at agent i. The logical time is defined to be the value of the clock immediately after the event. Before sending a message, the clock is first incremented and then attached to the message as a timestamp. When receiving a message, the local clock is increased to the greater of the received timestamp and the local clock after incrementing it by one. This new clock value is assigned to the receive event.

The final step is to add an algorithm to determine global properties. The idea is that all agents know about a predetermined logical time t and perform the following two steps: first, determine the agent state up to time t and after time t; second, determine the sequence of messages sent at logical times before t but received at logical times greater than t. The resulting state information represents an instantaneous global state snapshot and can be used to determine the progress or termination of the reconfiguration algorithm.

2) Chandy-Lamport Algorithm: The Chandy-Lamport algorithm [4] does not use an explicit logical time but rather uses "marker" messages to indicate the point in time that separates states and messages before and after the snapshot. The Chandy-Lamport algorithm augments the original reconfiguration algorithm A and yields the modified implementation $CL(A)$. An agent i may trigger the algorithm itself or be triggered through a "snap" message. After receiving the snap message, the agent records its state and sends "marker" messages to it neighbors. Afterward, it records the messages arriving to keep record of the channels' states until it receives markers for the respective channels. In case an agent receives a marker before it has recorded the state, it records its state, sends out marker messages, and starts recording incoming messages.

Snapshot algorithms can also be used for the purpose of detecting deadlocks in the reconfiguration algorithm as deadlock is a special case of a stable global property: a deadlock consists of a cycle of two or more processes, which wait for each other to release resources with no messages in channels connecting them.

C. Wave Algorithms

Wave algorithms [5] are computations eventually performed by every node using local information and information received from other agents. Many times wave algorithms are part of more complex algorithms to achieve the final algorithm. For example, waves may be used to broadcast messages in the distributed system [5] to "wake up" nodes to start processing data. A wave algorithm can be started by a fixed initiator or any arbitrary node in graph G. Once initiated by a node, the computation propagates throughout the system and returns to the initiator by a message passing scheme. When a node performs wave computations, it is "visited" by the wave. The termination detection algorithm based on waves propagates throughout the system through a variable "wave color." During the computation each node colors the wave black if the node is active and white if the node has been idle since the last visit of the wave. When the computation reaches the initiator it generates a "decide" event. If the color of the received wave is white then the decision is true otherwise it is false. If decision is true then algorithm detected termination, and if the decision is false then the node restarts the wave computation through another white wave.

IV. Main Algorithm

The termination detection algorithm for the shipboard power system is based on the wave algorithm concept introduced above and extended by features found in [9]. The network graph is divided into multiple rooted trees, and we chose the generator (source) nodes as root nodes. The wave algorithm is initiated by each root and spreads in downward direction to the leaves and then back upwards to the root. If an intermediate node receives the wave from all of its children then it sends the accumulated wave to its parent. Each node follows rules specified in the following paragraphs while sending the wave

signal. In this algorithm the wave computations do not terminate but continue forever to allow continuous detection and response to the changing environment of the reconfiguration algorithm.

The underlying construction of the spanning trees is based on breadth first search [8]. This algorithm forms multiple rooted trees at the start of the system and preserves these unless changes in physical connections occur. In this algorithm, each agent holds a unique parent node and a list of children nodes. A distance value (d-value) of a node is defined as the number of links from the root to the node. For the root node the d-value is always zero. Initially, the list of children and parent is empty and the d-values of all nodes except roots are set to a high number, e.g., total number of nodes in the system. Every node n compares the d-values of its neighbors and selects a node k with the lowest d-value d_k as its parent. It changes its local d-value to $d_k + 1$ and informs node k to update the list of children. Next, we define the sets wave colors C and agent states Z as

$$C \in \{Black\,(B),\, White\,(W)\}$$
$$Z \in \{Active\,(A),\, Idle\,(I),\, Terminated\,(T)\}$$

The color attribute of a wave will allow to determine the agents' states and consequently the detection of termination as follows. Any root starts the wave algorithm as soon as it receives the first child node by sending a white wave to it. When a wave visits an agent for the first time, the agent freezes its parent. If the agent has children then it forwards the wave to them, otherwise it sends it back to the parent. Thus eventually all nodes receive a wave, fix their parent and hold a list of children. When an agent freezes its position in a tree, it can start basic computations. For the remaining of the discussion the reconfiguration algorithm is referred to as basic computation and any reconfiguration message is termed a basic message. When the waves returns to the root, a new wave is started.

Any agent becomes active by execution of a basic computation or after receiving a black wave. An agent is idle when it has no basic computation to perform and forwards a white wave. Any node that receives a white wave consecutively two times detects global termination and executes any further actions required for system reconfiguration.

The following summarizes the rules defined for this algorithm.

Rules:

1) Every agent that receives a wave sends a $Black$ wave if the received wave is $Black$ or it currently performs any basic computation or received a basic message after sending the last wave.
2) Every agent that receives a wave sends a $White$ wave if the received wave is $White$ and it did not perform a basic computation or did not receive a basic message after sending the last wave.

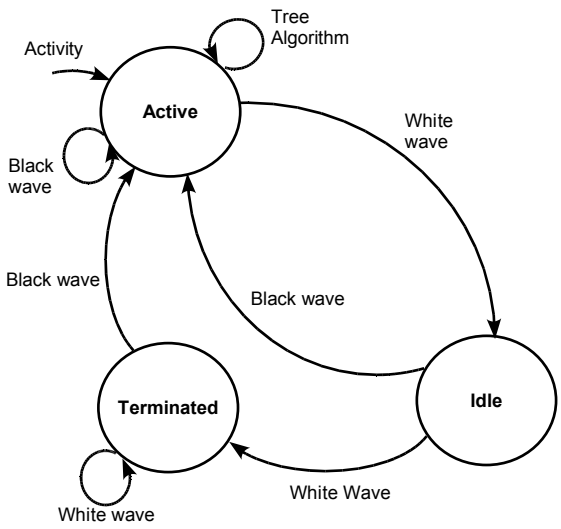

Fig. 3. State diagram, where the Black and White waves denote sending event

3) Any agent that sends a $Black$ wave becomes $Active$.

$$\begin{array}{ccc} Z & \Longrightarrow & A \\ \downarrow B & & \end{array}$$

4) Any $Active$ agent that sends a $White$ wave becomes $Idle$. The $Idle$ state signifies a partial termination. The node believes that all other agents have finished basic computation and it waits for the confirmation, i.e., another $White$ wave.

$$\begin{array}{ccc} A & \Longrightarrow & I \\ \downarrow W & & \end{array}$$

5) Any $Idle$ agent that sends a $White$ wave terminates.

$$\begin{array}{ccc} I & \Longrightarrow & T \\ \downarrow W & & \end{array}$$

6) Any intermediate agent that receives a wave from the parent sends out a wave to all its children by following the above rules.
7) Any intermediate agent that receives waves from all its children sends a wave to the parent by following the above rules.
8) Any leaf agent that receives a wave from its parent sends a wave to its parent following the above rules.
9) If a root receives a wave it follows the above rules to make a decision and change its status. If the root agent is active then it sends out a black wave, otherwise it sends a white wave.

TABLE I

AN EXAMPLE OF SPANNING TREES

Tree 1				Tree 2			
Agent	d-value	P	C	Agent	d-value	P	C
1	0		2	31	0		30
2	1	1	3,4,5	30	1	31	27,28,29
3	2	2		28	2	30	
4	2	2		29	2	30	
5	2	2	6,7	27	2	30	8,11
6	3	5	7,12	8	3	27	14
9	3	5	15	11	3	27	10,17
7	4	6	13	10	4	11	16
12	4	6	18	14	4	8	
15	4	9		17	4	11	20
18	5	12	21,24	16	5	10	
21	6	18		20	5	17	23,26
24	6	18		23	6	20	
13	5	7	19	26	6	20	
19	6	13	22,25				
22	7	19					
25	7	19					

P = Parent and C = Children

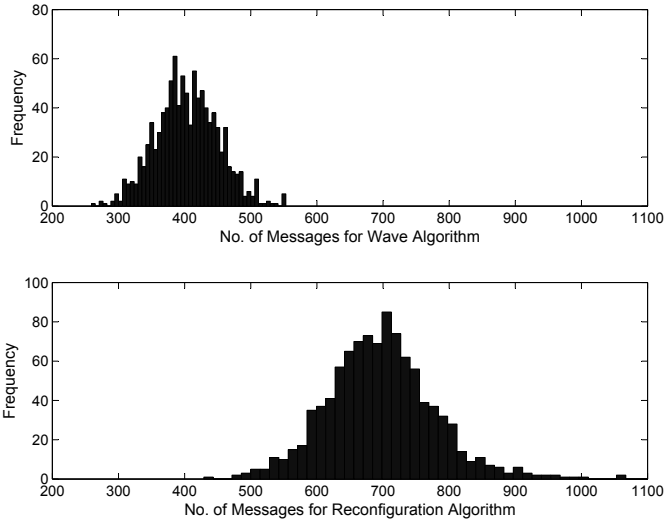

Fig. 4. Result showing frequency of number of messages required to complete the algorithm for startup case

Figure 3 shows the state diagram for the detection algorithm. Every agent follows these rules and processes the wave. Since the basic computation, i.e., the maximum flow algorithm is itself stable and eventually reaches termination, it is guaranteed that the wave algorithm will detect the termination.

V. DISCUSSION

Many of the above algorithms have one property in common, namely, the termination is detected by only one agent. As mentioned earlier for reconfiguration it is necessary to communicate the final termination decision to all agents in the system. This situation is comparable to one central process detecting termination and informing the other agents. This problem is addressed by the chosen wave algorithm through the propagation of two consecutive white waves and allow each node to locally decide on termination. The existence of multiple trees according to the number of available sources ensures that load demands are supplied as long as these are reachable from the remaining sources.

The wave algorithm removes the necessity of a blackboard in termination detection but some remarks with respect to underlying assumptions are in place. It is assumed that agents have been statically or dynamically assigned unique identifiers. The same applies to the list of neighbors known to an agent, which can also be achieved through a static or dynamic assignment. The distributed algorithm also relies on error free message delivery and the possibility to identify unresponsive agents. These issues are addressed in our approach through fixed agent identifiers, assignment of neighbors through the command and control center, and the message reliability of the Controller Area Network (CAN, [10]) and heartbeat messages, respectively. As the CAN is based on a bus topology, our testing of the physical system setup is restricted to failing agents only and failing links are modeled and tested through computer simulation. The wave forwarding mechanism can be applied to different network topologies but in our case we chose to use logical trees for routing messages. As the termination detection algorithm can restructure the trees when changes occur or agents fail to communicate, successful reconfiguration and termination takes place.

VI. EXAMPLE

The graph representing the shipboard power system testbed has been introduced in Fig. 2 above with 31 agents trying to find a solution. As two of the agents are source agents, the wave algorithm divides the system in two trees. An example of those trees is shown in Table I. The combined reconfiguration and termination detection algorithm is simulated and tested on a single desktop PC environment using MATLAB as programming environment. The list of reconfiguration scenarios studied includes initial system startup, changes in load demand, changes in edge capacity, loss of agents, and increases in demand beyond generation and power transfer capabilities. Statistics concerning the number of agents' moves necessary for the "startup" scenario where initially no power is provided to any load are given here. As the agents are randomly scheduled to perform their computations, 1000 runs were executed for each scenario and the found solutions tested for their validity. The criteria for success are: every run eventually converges and the power flows are according to load demand priorities and system capabilities. Figure 4 shows the number of messages required for the reconfiguration algorithm and wave algorithm for the startup case. In this scenario, the agents try to find a solution to a previously deenergized system. The average number of messages required for the wave algorithm is 392 and for reconfiguration algorithm 698.

VII. CONCLUSION

Several distributed algorithms are available for improving the reconfiguration algorithm of the energy management sys-

tem. The advantage of these approaches over the blackboard approach is the gained flexibility in detecting global termination of the agents algorithm through avoiding the centrally shared memory architecture. This paper presents one such algorithm applicable to reconfiguration systems and demonstrates a successful implementation.

Remaining challenges to the distributed energy management approach include the possibility of traitor agents. Such agents perform actions and send messages not according to the designed algorithms but try to fool other agents. Improvements necessary to allow at least a certain number of agents to behave as traitors is part of future research.

ACKNOWLEDGMENT

This research is sponsored in part by a grant from the US DEPSCoR and ONR (DOD/ONR N000 14-031-0660).

REFERENCES

[1] S. D. Pekarek, J. Tichenor, S. D. Sudhoff, J. D. Sauer, D. E. Delisle, and E. J. Zivi, "Overview of a naval combat survivability program," *Proceedings of the 13th International Ship Control Systems Symposium, Orlando, Florida*, 2003.

[2] A. Feliachi, K. Schoder, S. Ganesh, and H.-J. Lai, "Distributed control agents approach to energy management in electric shipboard power system," *Power Engineering Society General Meeting, 2006*, June 2006.

[3] F. Nissim, "Distributed termination," *ACM Transactions on Programming Language and Systems*, vol. 2, no. 1, pp. 42–55, Jan 1980.

[4] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 2003.

[5] G. Tel, *Introduction to Distributed Algorithms*. Cambridge University Press, 1994.

[6] E. Godard, Y. Metivier, M. Mosbah, and A. Sellami, "Termination detection of distributed algorithms by graph rebelling systems," *International Conference on Graph Transformations 2002 (ICGT), Springer-Verlag Berlin Heidelberg*, pp. 106–119, 2002.

[7] P. Pant, K. Schoder, and A. Feliachi, "On electric shipboard power system testbed," *Proceedings of the IEEE Electric Ship Technologies Symposium, ESTS 2007, Arlington, Virginia, USA*, 2007.

[8] S. Ghosh, A. Gupta, and S. V. Pemmaraju, "A self-stabilizing algorithm for the maximum flow problem," *Computers and Communications, 1995. Conference Proceedings of the 1995 IEEE Fourteenth Annual International Phoenix Conference on*, pp. 8–14, Mar 1995.

[9] N. Francez and R. Michael, "Achieving distributed termination without freezing," *IEEE Transactions on Software Engineering*, vol. SE-8, no. 3, pp. 287–292, May 1982.

[10] R. Bosch GmbH, *CAN Specification 2.0*, 1991. [Online]. Available: http://www.semiconductors.bosch.de/

Pinak Tulpule (M'2003) is a graduate research assistant at APERC and pursuing his Masters degree in Electrical Engineering at the Lane Department of Computer Science and Electrical Engineering at West Virginia University. He completed his Bachelor degree in Electronics and Telecommunication from Pune University, India, in July 2004. His interests include software agents, neuro-fuzzy control, and their application to energy management systems.

Karl Schoder (M'2000) received his Diplom-Ingenieur in Electrical Engineering from the University of Technology Vienna in 1997 and his Ph.D. from West Virginia University in 2002. He has been working on modeling and simulating electric power systems and FACTS devices. His research interests include modeling, analysis and control of electric transmission and distribution systems.

Ali Feliachi (M'83, SM'86) received the Diplôme d'Ingènieur en Electrotechnique from Ecole Nationale Polytechnique of Algiers in 1976, and MS (1979) and Ph.D. (1983) in EE from Ga Tech. He is the holder of the Electric Power Systems Chair Position, and the Director of the Advanced Power & Electricity Research Center at West Virginia University. He has been working in the field of large scale and power systems for 30 years and has over 200 publications.

Hong-Jian Lai received his Ph.D. in Mathematics from Wayne State University in 1988. He is a full Professor of the Department of Mathematics at West Virginia University. He has been working on discrete mathematics, algorithm and optimization for 15 years.