Distributed Control Agents Approach to Energy Management in Electric Shipboard Power Systems

Ali Feliachi, *Member, IEEE,* Karl Schoder, *Member, IEEE,* Shilpa Ganesh, and Hong-Jian Lai

***Abstract*— The automatic reconfiguration of electric shipboard power systems is an important step toward improved fight-through and self-healing capabilities of naval warships. The improvements are envisioned by redesigning the electric power system and its controls. This paper presents a new scheme for an energy management system in the form of distributed control agents. The control agents' task is to ensure supply of the various load demands while taking into consideration system constraints and load priorities. A graph theoretic self-stabilizing maximum flow algorithm for the implementation of the agents' strategies has been developed to find a global solution using local information and a minimum amount of communication. A case study using the distributed agents within a multilayer system architecture to function as energy management system is presented.**

***Index Terms*— integrated electric power systems, multi-agent systems, autonomous energy management**

I. Introduction

The success of increasingly all electric commercial ships has sparked interest to incorporate the advantages of improved electric power systems in naval warships. Design of shipboard power generation, distribution, communication, propulsion, and control is dictated by the demand to reduce manning and costs and is step-by-step replacing mechanical-hydraulic systems by electric ones. This process will ultimately also improve the survivability of the envisioned all electric warship.

The Naval Combat Survivability generation and propulsion and DC distribution testbeds have been developed as a common tool for modeling the electric warship [1], [2]. The remaining control challenge requires advanced control strategies to ensure system stability and to take advantage of the system's capabilities [3]. The controls to be developed include local controls for the various power electronic devices and the automation of processes to help system operators cope with routine operations, maintenance, and emergency situations to optimize lifetime and cost performance. The necessity of improving warships is the bottom line of several incidents in which the electric power was lost through only a single incident [4]. The answer to this problem is envisioned in dependable automation strategies [5] that for example extend nonlinear control theory, apply analytic redundancy, utilize distributed intelligence 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 of the electric shipboard power system is addressed here.

The energy management system proposed in this paper is to distribute the available power to the loads. Under normal conditions all the loads are supplied, but in case of a disturbance that results in reduced power supply only the high-priority or critical loads are served. This part of the problem is usually known as the power flow problem and can be solved at a central place numerically using various techniques, for example Newton-Raphson algorithm to solve the nonlinear set of equations and it can also be formulated as linear programming problem to allow incorporation of limits and priorities of loads and supply paths. These approaches have been investigated by several authors and can be found in [6] and [7]. A different approach in solving the power flow problem for the energy management system is taken here by using spatially distributed control agents. These software agents make local decisions to reach a globally acceptable solution with limited amount of communication.

The use of software agents gained popularity by recent advances in software engineering to construct distributed systems that cooperate to reach a common goal [8]. This new philosophy to implement decentralized control algorithms is discussed in this paper and envisioned improvements and advantages of a multi-agent based control framework are applied to build the energy management system. Parts of the distributed approach have been presented in [9], [10], this paper expands on the control agents' algorithm and case study used to test convergence characteristics.

The distributed agent concept needs to be complemented by an appropriate framework for implementing automatic reconfiguration. Graph theory provides a formal basis to represent the distributed control system and to develop algorithms for a decentralized energy management solution. Maximum flow algorithms have been implemented to find an answer to this challenge.

The paper is organized as follows. In the next section an introduction to the electric shipboard power system as the domain of application is presented. Then the proposed multi-agent system architecture is described followed by an overview of graph theory and the maximum flow problem that is used to design the energy management system. Finally, a case study to demonstrate the distributed and agent-based concept is given.

This research is sponsored in part by a 2003 grant from the US DEPSCoR and ONR (DOD/ONR N000 14-031-0660).

The authors are with the Advanced Power and Electricity Research Center, West Virginia University, Morgantown, WV 26506–6109, Email: alfeliachi@mail.wvu.edu, karl.schoder@mail.wvu.edu, hjlai@math.wvu.edu.

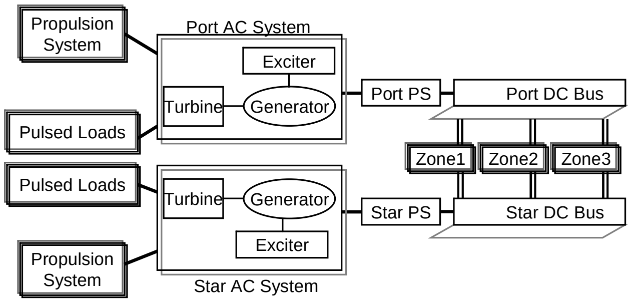


Fig. 1. Shipboard power system overview

II. System Description

The system used in the case studies has been described in detail in [1] and [2]. A simplified representation of its components has been implemented to avoid excessive simulation times due to unnecessary details concerning the power electronic components. An overview of this system in form of a block diagram is given in Figure 1. The distribution testbed is fed by two AC sources represented by gas turbines, synchronous generators, and exciters. The propulsion systems are connected to the AC system using power electronics for a flexible drive system. The AC is converted to DC for distribution of the electric power by two DC buses, the port bus and starboard bus. To improve the survivability, the loads on board are grouped into zones and each zone is supplied from both DC buses by additional DC/DC converters. This allows a desireable redundancy as well as a graceful performance degradation as the converters are used to actively limit currents and can be used to isolate faulty loads or entire parts of the system. Also, the converters can be operated to supply loads from either or both buses and to switch continuously between them. The loads encountered represent typical power demands including induction motors, three-phase AC-loads, and constant power loads.

All components are modeled by transfer functions of first- or second-order to represent input-output relationships relating active power and voltage values. The components' transfer functions and parameters are given in [9], [10]. The following sections discuss the agent based framework and the maximum flow algorithm used to create the energy management system.

III. Multi-Agent Systems

A single agent is a system that is situated in some environment and capable of taking autonomous actions in this environment in order to meet its design objectives [8]. Agents are capable of flexible autonomous actions through reactivity – responding in a timely manner, pro-activity – taking initiative through goal-directed behavior, and social ability – interacting. A multi-agent system is composed of a population of agents, which interact with each other to reach common objectives, while simultaneously each agent pursues individual objectives [11].

The agents form a loosely-coupled network of problem solvers and work together to solve problems that are beyond their individual capabilities. Together, the agents increase fault-tolerance as inherently distributed mechanism and thus a system made of autonomous agents will not collapse when one or more of its components fails. This modular approach creates a scalable architecture and agents become powerful entities because of the factorization of the problem they provide. Each agent can be identified as an entity (e.g., a machine, a plant, a tool or a part) and thus help in incremental growth and flexible expansion. The advantage of scalability is provided as each agent can join a system, starts working with other agents, or just leaves a system once it has finished a plan it was engaged in without effecting the operation of the system.

Agents have the capability to reconfigure themselves. This is an important advantage for systems that must respond to a wide range of different conditions. Because each agent is in close contact with the real world, the system's computational state tracks the state of the world closely. As the overall system behavior emerges from local decisions, the system readjusts itself automatically to changes in the environment or the removal of other agents. Thus a fully functional self-configuring system can be effectively implemented by merely networking agent-based resources.

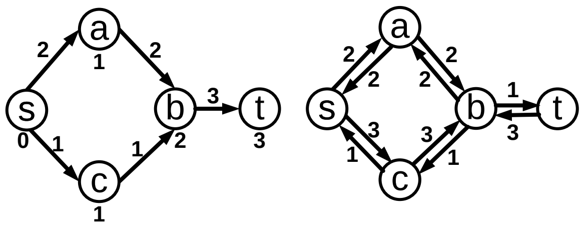
The communication among agents can be achieved by different means. The work presented in this paper makes use of the blackboard architecture as described in [8]. The blackboard is, for example, used to notify other agents of their current status with respect to being active making local changes or running idle.

IV. Maximum Flow Problem

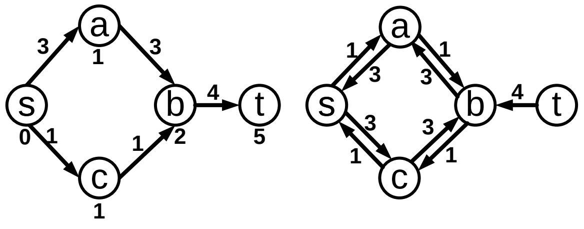
A fundamental problem in graph theory is the maximum flow problem for which many different sequential and parallel algorithms have been developed [12]. All the parallel algorithms but the one in Gosh, Gupta, and Pemmaraju [13] present implementations that do not run in polylogarithmic time on a polynomial number of processors. Other advantages of this algorithm are its simplicity, passiveness, self-stabilizing, and local checking and corrections only. This algorithm has been adapted here to find a solution to the power flow problem of the electric shipboard system. The following briefly defines the maximum flow problem and notation used. More details can be found in [13].

A directed graph (digraph) $G = (V, E)$ with $n = |V|$ number of nodes and $e = |E|$ number of edges is shown in Figure 2. The graph should not contain any directed cycles of length 2: if edge $(i,j) \in E$ then $(j,i) \notin E$. The nodes s and t are the distinct source and sink nodes, respectively. The source node s has no incoming edges, the sink node t has no leaving edges. Every edge (i,j) is associated with a real-valued flow $f(i,j)$ and a non-negative real-valued capacity $C(i,j)$. Any feasible flow f in G obeys the flow conservation constraint, i.e., the incoming flow $I_f(i) = \sum_{(j,i)\in E} f(j,i)$ equals the outflow $O_f(i) = \sum_{(i,k)\in E} f(i,k)$: for all $i \in V - \{s,t\}$: $I_f(i) = O_f(i)$. Flows are also satisfying the skew symmetry property of $f(i,j) = -f(j,i)$. The maximum flow problem is to maximize the outflow of the source node $O_f(s)$.

Definition 1: For any flow f and for each pair of nodes $(i,j) \in V \times V$, the residual capacity $r(i,j)$ is equal to $C(i,j) - f(i,j)$.



(a) Feasible flow and its residual graph



(b) Maximum flow and its residual graph

Fig. 2. Example graph with edge capacities of 4.

Definition 2: For any flow f in G, the residual graph G_f of G with respect to f is defined as the weighted digraph $G_f = (V_f, E_f)$, where $V_f = V$ and $(i, j) \in E_f$ if and only if $r(i, j) > 0$.

Definition 3: A directed path in the residual graph G_f from s to t is called an augmenting path.

Example: A feasible flow of a graph and its residual graph are given in Figure 2(a), where the directed graph is shown on the left and its residual graph on the right. All edges have a capacity of 4 and $C(s, a) = C(a, b) = C(s, c) = C(c, b) = C(b, t) = 4$. The number on an edge indicates the flow along an edge, e.g., the flow from node b to the target node t is $f(b, t) = 3$. The residual graph for the flow gives the remaining capacity, for example the residual flows $r(b, t) = C(b, t) - f(b, t) = 4 - 3 = 1$ and $r(t, b) = C(t, b) - f(t, b) = 0 - (-3) = 3$. A maximum flow and its corresponding residual graph are shown in Figure 2(b), respectively. It has been achieved by increasing the flow on the augmenting path $s - a - b - t$ by its minimum residual capacity of $r(b, t) = 1$. Note that after increasing the flow, the updated residual graph contains no augmenting path and, therefore, the flow from the source to the target cannot be increased further.

Definition 4: The distance value $d(i)$ of a node denotes the "believed" length of the shortest directed path from s to i in the residual graph G_f. The value of $d(i)$ is restricted to be in the range $[0 \dots n]$ because the longest distance in a graph G is $n - 1$. The special values of 0 and n indicate the source's distance value and a node with no directed incoming path in the residual graph, respectively.

Example: The d-values for the previously discussed example graph are the numbers below the nodes. Initially, all the d-values represent the distance to the source node. But as the maximum flow is achieved, the target's d-value changes to the maximum possible value of $n = 5$ and indicates that no additional demand can be satisfied.

V. Shipboard's Power Flow Problem as a Maximum Flow Problem

The electric shipboard power system is modeled as an acyclic directed graph and its residual graph. The transformation of the block diagram into the acyclic graph is achieved by representing physically linked components by neighboring nodes in a graph. Each of the edges between nodes can be associated with a direction because the electric power flows only from the AC generators toward the loads. The multiple generators and loads can be accommodated by introducing additional source and sink nodes where the edges' capacities from the source and to the sink nodes represent the upper bounds on generation capabilities and load demands, respectively. Each agent should perform locally by observing its environment and taking corrective actions to satisfy its goals, i.e., request power for a load, route the power flow according to load priorities, etc. The moves themselves are implemented by the agents without synchronization. By making local and asynchronous moves that follow a self-stabilizing concept, convergence to the desired solution is achieved. Also, as disturbances can be seen as arbitrary initial conditions, tolerance to transient faults that upset system conditions is given. The algorithm published by [13] provides such a framework for the desired agents' behavior. It allows the design of agents that adjust to dynamic changes in network topology and edge capacity. Its computational model and changes made to accommodate requirements of the shipboard power system are presented next.

A. Computational model

Each edge (i, j) corresponds to a physical link and each node i in graph G is represented by an agent i. Every agent i contains a set of local variables and executes a program (algorithm) asynchronously that can be expressed by the conditional execution of actions:

```
do
   $G_1$ ? $A_1$
   $G_2$ ? $A_2$
     ...
end;
```

Each action A_i modifies the agent's local variables. The guards G_i are boolean functions of agent i's variables. If more than one guard is activated then only one is randomly chosen and its corresponding action executed. Afterwards the cycle begins again by evaluating the guards before an action is chosen.

B. Representation of the Electric Shipboard Power System as Graph and Terminology Used

This section discusses the translation of the physical system into its equivalent graph, definitions, and terminolgy used. Node and agent types: source nodes, router nodes, load nodes, sink nodes. Source and sink nodes have been introduced above, and the meaning of the other two is as follows. Router nodes represent devices that distribute power along outgoing connections. Every load node carries additional information concerning its priority, has only one outgoing link, and this outgoing link is toward a sink node. Load node agents use their priority and load demand to set their respective link information that consequently propagates through the network.

Besides the d-value and flows, agents need to keep track of priority information. Therefore, each link carries the addtional information in form of the following quadrupel: supplied load, supplied load priority, unsupplied load, and unsupplied load priority. These values are updated by the agents. A second addition to the standard maximum flow algorithm is the dL-value. The dL-value is based on the d-value idea but measures the distance from the sink node rather than from the source node. The following describes the agents' algorithm.

C. Agents' Maximum Flow Algorithm

The agents represent the nodes in the acyclic digraph G. The flow on the edges between nodes i and j is stored in variable $f(i,j)$. Two agents are neighbors in case an edge (i,j) exists. Only neighboring agents communicate to solve the maximum flow problem. Each agent can update the flow by directly modifying $f(i,j)$ and keeps track of its believed distance values $d(i)$ and $dL(i)$ to the source and load, respectively. The priority related information concerning supplied and unsupplied demand along links (i,j) is denoted by $(f_{SD}(i,j), p_{SD}(i,j), f_{UD}(i,j), p_{UD}(i,j))$ and propagated by each agent from the outgoing links to the incoming links.

Every agent in the graph is allowed to make moves by changing its distance values, flows, and priorities. The algorithm executed by the agents at their nodes i $(i \neq s)$ is based on nine guarded statements GS_1–GS_9 that consist of their respective guard G_i and action A_i. The first eight actions represent local steps taken to find a solution to the power flow problem and notification of the blackboard of the agent's busy-status in case it has been idle before. Guarded statement GS_9 notifies other agents by publishing to the blackboard the agent's convergence to a local solution.

The algorithm is continuously executed by every agent $i \neq s, t$ in order to restore the $demand(i) = O_f(i) - I_f(i) = 0$. The two distinguished agent types s (source) and t (sink) differ in their behavior from other nodes: Agent s is idle; agents t performs the same program as the other nodes but with a $demand(t) = \infty$, leading the agents towards a maximum flow solution. The agents' actions perform changes to increase or decrease their inflows or to reduce their outflows. Breadth-first search is used to update the believed shortest distance values $d(i)$ and $dL(i)$. An agent with the belief that no direct path from the source to itself exists pushes back the demand on an outgoing edge. The agents are able to differentiate among load prioritites by keeping track of the supplied and unsupplied demand and their respective priorities along links. The following describes the guarded statements (GS) in more detail:

GS$_1$ – update d-value: Every agent i, $i \neq s$, updates its $d(i)$-value by determining $d(k) = \min\{d(j)|(j,i) \in E_f\}$. If $d(k) < n$ then node k is a predecessor of node i and $d(i) = d(k) + 1$, otherwise $d(i) = n$ and no direct path from any source to node i exists. The distance value of the source node is set to $d(s) = 0$. If the agent has been previously idle then the blackboard is informed of the agent's busy-status.

GS$_2$ – demand<0: One of the following two situations can occur:

Unsupplied load: An agent i, $i \neq s, t$ uses the excess to supply the unsupplied demand $f_{UD}(i,j) > 0$ and $p_{UD}(i,j) = \max\{p_{UD}(i,k)|k \in Ef\}$ through increasing the flow along path (i,j) by $f_{inc} = \min(-demand, r(i,j))$ and sets $f_{UD}(i,j) = f_{UD}(i,j) - f_{inc}$.

Excess supply: Every agent $i, i \neq s$, reduces the flow along the lowest priority incoming edge (j,i) in case $demand(i) < 0$ by $\min(-demand, f(j,i))$, where $p(j,i) = \min\{p(k,i)|(k,i) \in Ef\}$.

In both cases, if the agent has been previously idle then the blackboard is informed of the agent's busy-status.

GS$_3$ – increase inflow: Every agent i, $i \neq s$, with $demand(i) > 0$ and $d(i) < n$ increases the flow on the highest priority and available incoming path $(j,i) \in E_f$ by the minimum of the demand and residual flow $r(j,i)$. A sink agent also updates information concerning supplied and unsupplied demand: $f_{SD}(j,i) = f(j,i)$ and $f_{SD}(i,j) = C(j,i) - f(j,i)$. If the agent has been previously idle then the blackboard is informed of the agent's busy-status.

GS$_4$ – push back demand: Every node i, $i \neq s$, with $demand(i) > 0$ and $d(i) = n$ reduces the outflow towards the lowest priority load along path $(i,j) \in E_f$, where $p_{SD}(i,j) = \min(p_{SD}(i,k)|k \in Ef)$, by $f_{dec} = \min\{demand(i), f(i,j), f_{SD}(i,j)\}$. The demand information of the outgoing link is updated accordingly: $f_{SD}(j,i) = f_{SD}(j,i) - f_{dec}$. If the agent has been previously idle then the blackboard is informed of the agent's busy-status.

GS$_5$ – remove capacity violation: Every node i, $i \neq s$, checks the flows on incoming edges $(j,i) \in E$ and reduces flows in case of capacity violations. Note that the other guarded statements never lead to this condition in case a valid initial flow existed. This statement rather allows changing system conditions including the loss of a link. If the agent has been previously idle then the blackboard is informed of the agent's busy-status.

GS$_6$ – priority update: Every node i, $i \neq s, t$ with $demand(i) = 0$ updates incoming edges' $(k,i) \in E$ priority information according to the lowest priority value of supplied load and highest priority unsupplied load along outgoing edges $(i,j) \in E$. Each of the incoming edges information is set to the following quadruple: flow towards minimum priority supplied load, priority of minimum priority load, unsupplied load demand, priority of unsupplied load demand). The priority information is determined by $(f_{SD}(i,j), p(f_{SD}(i,j) = \min\{p(f_{SD}(i,k)|k \in Ef)\}, f_{UD}(i,l), p(f_{UD}(i,l)) = \max\{p(f_{UD}(i,k)|k \in E\}$. This statement ensures that higher priority loads will be preferred throughout the system and the lowest priority demand reduced first. If the agent has been previously idle then the blackboard is informed of the agent's busy-status.

GS$_7$ – reroute outflow: Every agent i, $i \neq s$, with $demand(i) = 0$ and $d(i) = n$ checks for unsupplied load demands $f_{UD}(i,j)$ and $dL(j) < n$ along its outgoing edges and diverges outflow from lower priority loads k to the highest priority unsupplied load j: $f_{inc} = \min(f_{UD}(i,j), r(i,j), f(i,k))$. The new flows are given by

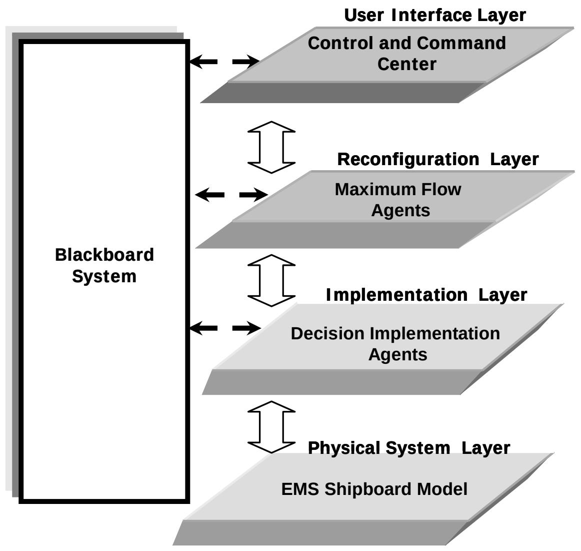


Fig. 3. Layered architecture for the energy management system

$f(i,j) = f(i,j) + f_{inc}$ and $f(i,k) = f(i,k) - f_{inc}$. The priority related information is adjusted to $f_{UD}(i,j) = f_{UD}(i,j) - f_{inc}$. If the agent has been previously idle then the blackboard is informed of the agent's busy-status.

GS$_8$ – update dL-value: Every agent i, $i \neq s,t$, updates its $dL(i)$-value by determining $dL(k) = \min\{d(j)|(i,j) \in E_f$. If $dL(k) < n$ then node k is a predecessor of node i towards a sink node and $d(i) = d(k) + 1$, otherwise $d(i) = n$ and no direct path from node i to a sink exists. The dL-value of a sink node is set to $dL(t) = 0$. If the agent has been previously idle then the blackboard is informed of the agent's busy-status.

GS$_9$ – publish status: Every node i, $i \neq s$, publishes its status of convergence to local a solution to the blackboard.

VI. Implementation

The energy management system for the electric shipboard power system has been accomplished following the agent-based framework. The different system parts have been grouped into layers to form a logical architecture according to their functionality (see Figure 3.) The top layer represents the human-machine interface for the command and control center. This layer displays important system information and allows the system operator to communicate with system components, e.g., request for desired changes in propulsion power. It can also be used to simulate disturbances, e.g., loss of generation.

The decentralized maximum flow algorithm is part of the reconfiguration layer. The agents execute their programs asynchronously and locally but can communicate status and results (internal variables) using the blackboard. The implementation layer represents the decision implementation agents. These agents are notified when a new power flow solution has been agreed on by the agents of the reconfiguration layer. These agents access locally their respective maximum flow agents to inquire the solution to be implemented. The lowest layer represents the mathematical model for the physical system. The shipboard model described earlier simulates the various components of the ship and their interactions. The components have local controls, e.g., proportional-integral controllers and their reference and parameter values, which are modified and adjusted by the decision implementation agents.

VII. Case Studies

The translation of the shipboard power system into its equivalent graph for the energy management system requires 30 agents – one agent for each of the components and sink agents for the eight loads (see Tables I and II.) The demand and capacity values have been per unitized and the smallest change in flows is 1 pu. This setup has been tested for different conditions, e.g., initial startup, changes in load demand, changes in edge capacity, loss of agents (nodes), and increases in demand beyond generation and power transfer capabilities. Three of these reconfiguration scenarios will be presented in the following: startup (Case A), loss of a link (Case B), and change in a load's priority (Case C).

TABLE I
Data for shipboard power system agents

No.	Component	Name	Type	Demand	Priority
1	AC generator port	AC2	Source		
2	AC bus port	BA2	Router		
3	Propulsion port	P2	Load	12	0
4	Propulsion port		Sink		
5	AC gen. starboard	AC1	Source		
6	AC bus starboard	BA2	Router		
7	Prop. starboard	P1	Load	12	0
8	Prop. starboard		Sink		
9	PS port	S2	Router		
10	PS starboard	S1	Router		
11	DC bus port	BD2	Router		
12	DC bus starboard	BD1	Router		
13	CM Z1 port	C4	Router		
14	CM Z2 port	C5	Router		
15	CM Z3 port	C6	Router		
16	CM Z1 starboard	C1	Router		
17	CM Z2 starboard	C2	Router		
18	CM Z3 starboard	C3	Router		
19	Load1 Zone1	L1	Load	3	2
20	Load1 Zone1		Sink		
21	Load2 Zone1	L2	Load	2	1
22	Load2 Zone1		Sink		
23	Load1 Zone2	L3	Load	1	2
24	Load1 Zone2		Sink		
25	Load2 Zone2	L4	Load	1	1
26	Load2 Zone2		Sink		
27	Load1 Zone3	L5	Load	1	2
28	Load1 Zone3		Sink		
29	Load2 Zone3	L5	Load	1	1
30	Load2 Zone3		Sink		

To be able to test convergence characteristics, the agents are executed in random order but each agent is executed only once per negotiation round. This process is repeated until a global solution has been found, i.e., one run is completed. Convergence can be determined when none of the agents makes any move, i.e., makes a change to distance or flow values associated with the guarded statements 1–8.

Case A - Startup: This scenario tries to find a solution to an initially deenergized electric power system. According to the data, the expected solution is as follows: All but one loads can be fully supplied. Load 2 in Zone 1 is curtailed

TABLE II
DATA FOR SHIPBOARD POWER SYSTEM LINKS

No.	From Agent	To Agent	Capacity	Priority
1	1	2	18	1
2	2	3	12	1
3	3	4	12	1
4	5	6	18	1
5	6	7	12	1
6	7	8	12	1
7	2	9	6	1
8	6	10	6	1
9	9	11	6	2
10	10	11	6	1
11	10	12	6	2
12	9	12	6	1
13	11	13	2	1
14	11	14	2	1
15	11	15	2	1
16	12	16	2	1
17	12	17	2	1
18	12	18	2	1
19	13	19	2	1
20	16	19	2	1
21	19	20	2	1
22	13	21	2	1
23	16	21	2	1
24	21	22	2	1
25	14	23	2	1
26	17	23	2	1
27	23	25	2	1
28	14	25	2	1
29	17	25	2	1
30	25	26	2	1
31	15	27	2	1
32	18	27	2	1
33	27	28	2	1
34	15	29	2	1
35	18	29	2	1
36	29	30	2	1

TABLE III
NUMBER OF MOVES (GUARDED STATEMENTS 1–8) TAKEN BY AGENTS (1000 RUNS)

	Case A: Startup	Case B: Loss of Link	Case C Priority
Max	258	160	165
Avg	216	44	56
Min	179	27	19
Std.	13	20	23

TABLE IV
NUMBER OF MOVES FOR DIFFERENT SOLUTIONS FOUND (1000 RUNS)

Case A					
Solution	Occurance	Min	Mean	Max	Std.
1	71	195	220	252	12
2	918	179	216	258	13
3	7	205	218	232	11
4	2	219	227	235	11
5	1	223	223	223	0
6	1	213	213	213	0
Case B					
Solution	Occurance	Min	Mean	Max	Std.
1	877	27	43	139	20
2	116	37	44	82	6
3	1	90	90	90	0
4	6	80	104	160	28
Case C					
Solution	Occurance	Min	Mean	Max	Std.
1	27	75	84	96	5
2	783	22	52	165	21
3	57	19	27	34	3
4	133	56	83	160	13

due to its lower priority to half of its requested demand. The maximum flow problem has been solved 1000 times to allow estimation of upper and lower bounds on negotiation rounds and moves necessary. Figure 4(a) shows the number of moves made in each negotiation round and the histogram concerning the negotion rounds necessary before a solution is found. A maximum of 25 negotion rounds is necessary to solve the problem. The number of active moves (as related to guarded statements 1–8) for each of the solutions is depicted in Fig. 4(b). Information concerning the type of move made (guarded statements chosen) during the negotiation process can be seen in Fig. 4(c): “active” represents guarded statements 1–8 and “idle” represents guarded statement 9. Maximum, average, minimum, and standard deviation for the number of agents’ moves made are given in Table III. As the solution to this maximum problem is not unique but rather 45 different solutions exist, the agents find 6 of these as the algorithm considers link priorities rather than randomly chosing an available link. Table IV gives details concerning the moves for these 6 solutions. The solution most often found (solution 2) is depicted Fig. 7(a).

Case B - Loss of a link: The second scenario concerns rerouting of energy due to a loss of a power supplying link. The chosen initial operating condition is based on the solution most often found to the above discussed startup scenario (Fig. 7(a)). Edge $(S2, BD2)$, which represents the link between the Port PS and Port DC bus, is supplying most of the power to the zonal loads. The loss of this edge, which is analogous to either loss of communication between the agents or a physical cable damage, was triggered through the command and control center. For this condition, the agents in the reconfiguration layer pick up the disturbance and start to negotiate a new globally acceptable solution. Once the agents agree upon a solution, a new supply path is found and the loads can be restored. For this scenario, edges $(S2, BD1)$ and $(S1, BD2)$ are now supplying part of the demand (Fig. 7(b)). Figure 5(a) shows the number of moves made by the maximum flow agents when reconfiguring the shipboard power system after loosing the link. Table IV gives details concerning the moves for the 4 solutions found in 1000 runs. Though the number of moves is lower than in the startup scenario, the information propagates slower through the network and both the number of negotiation rounds and standard deviation are increased.

Case C - Priority change: The third scenario investigates the change in a load’s priority. Again, the starting point is the solution found in the startup scenario as descript above (Fig. 7(a)). The change concerns the previously low priority load in zone 1 that is made the higher priority load. The agents pick up this incremental change and start negotiating a new solution. Figure 6(a) shows the number of moves made by the maximum flow agents when reconfiguring the shipboard power system. Table IV gives details concerning the moves for the 4 solutions found in 1000 runs. Again, though the number of moves is less than in the startup scenario, the number of negotiation rounds and standard deviation are increased. Also, two distinct peaks appear in the number of negotiation rounds (9 and 16) and moves taken (40 and 75). The solution most often found (solution 2) is depicted Fig. 7(c). As expected, load 1 is now only partly supplied in favor of load 2.

VIII. Discussion

The distributed agent concept find a solution to the energy management problem. Priorities have been introduced primarily due to the Navy's needs in distinguishing the components' priorities. The priorities can be adjusted by agents according to their valuation of being supplied or not. Priorities may also represent specific contracts between suppliers and consumers that would allow apply this concept to public distribution systems given an appropriate level of observability and controllability of power flows.

A blackboard with a rather inflexible communication scheme was implemented here to allow the agents to communicate and to proof the concept. A more formal agent communication language could be defined to achieve inter-agent communication following any of the currently available recommendations, e.g., based on the Foundation for Intelligent Physical Agents (FIPA) concept. Also, the centralized blackboard scheme represents a bottleneck and a possible reliability risk.

Solutions to the reconfiguration problem have been found in 0.5 to 1.5 seconds on a single desktop PC implementation using Matlab as programming environment and a general purpose operating system. Therefore, more stringent real-time requirements should be achievable even in a distributed environment due to a more deterministic and dedicated processing and communication infrastructure. Currently, a small prototype implmentation of the shipboard power system with a Controller Area Network based communication is developed to allow real-time performance tests.

IX. Conclusion

A new agent-based concept for the automatic reconfiguration of the electric shipboard power systems has been presented. A multilayer architecture has been used to accomplish the functionality of a human-machine interface, automatic reconfiguration using a maximum flow algorithm, agents to implement reconfiguration decisions, and the mathematical model of a shipboard power system. Most of the actions necessary for the operation of the energy management system can be performed locally and, therefore, only a limited amount of global communication is required. The case study demonstrated the feasibility and flexibility of the concept and results are promising. The agents are able to adjust to changes in the system and allow the automation of actions usually performed by humans and converge to a solution with quality attributes according to given priorities.

References

[1] Pekarek, S.D., Tichenor, J., Sudhoff, S.D., Sauer, J.D., Delisle, D.E., Zivi, E.J., "Overview of a Naval Combat Survivability Program," *Proc. 13th Int. Ship Control Systems Symposium*, Orlando, FL, 2003.

[2] Sudhoff, S. D., Glover, S. F., Zak, S. H., Pekarek, S. D., Zivi, E. J., Delisle, D. E., "Stability Analysis Methodologies for DC Power Distribution Systems," *13th International Ship Control Systems Symposium*, Orlando, FL, USA, April 7–9, 2003.

[3] Zivi, E. L., McCoy, T. J., "Control of a Shipboard Integrated Power System," *Proceedings of the 33rd Annual Conference on Information Sciences and Systems*, Baltimore, MD, 1999.

[4] Tucker, A. J., "Opportunities and Challenges in Ship Systems and Control at ONR," *IEEE Conf. on Decision and Control*, Dec. 4, 2001.

[5] US Navy ONR/NSF EPNES, "ONR Control Challenge Problem (white paper)," http://www.usna.edu/EPNES/Challenge Problem.htm, 2002.

[6] Butler, K. L., Sarma, N. D. R., Prasad, V. R., "Network reconfiguration for service restoration in shipboard power distribution systems," *IEEE Transactions on Power Systems*, vol. 16, pp. 653–661, November 2001.

[7] Butler-Purry, K. L., Sarma, N. D. R. , "Self-Healing Reconfiguration for Restoration of Naval Shipboard Power Systems," *IEEE Transactions on Power Systems*, vol. 19, no. 2, pp. 754–762, May 2004.

[8] Weiss, G., *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999.

[9] Ganesh, S., Schoder, K., Lai, H. J., Al-Hinai, A., Feliachi, A., "Energy Management System with Automatic Reconfiguration for Electric Shipboard Power Systems," *Proc. of ASNE Reconfiguration and Survivability Symposium, RSS 2005*, Jacksonville, Florida, February 2005.

[10] Ganesh, S., *Mulitagent Autonomous Energy Management*, MSEE thesis, Lane Department of Computer Science and Electrical Eng., West Virginia Univerisity, November 2005.

[11] Ferber, J., *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison–Wesley, Harlow, UK, 1999.

[12] Ahuja, R. K., Magnanti, T. L., Orlin, J. B., *Network Flows - Theory, Algorithms, and Applications*, Prentice Hall, 1993.

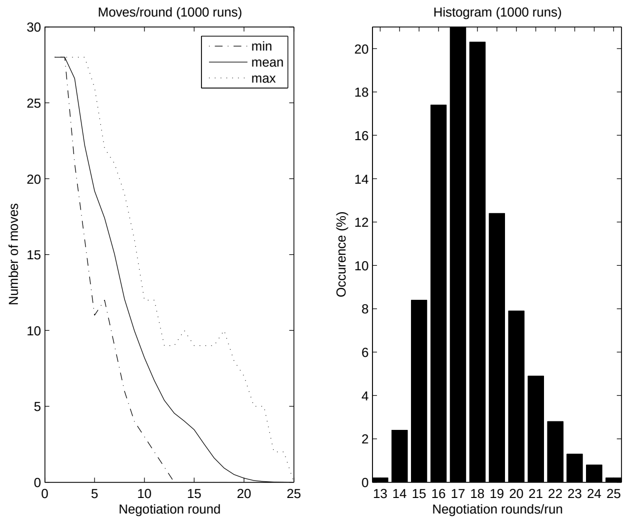
[13] Gosh, S., Gupta, A., Pemmaraju, S. V., "A Self-stabilizing Algorithm for the Maximum Flow Problem," *Distributed Computing*, vol. 10, 167–180, 1997.

Ali Feliachi received a Diplôme d'Ingénieur en Electrotechnique from the Ecole Nationale Poltytechnique of Algiers, Algeria, in 1976, and MS (1979) and Ph.D. (1983) in Electrical Engineering from Ga Tech. He is a Full Professor and the holder of the Electric Power Systems Chair position in the Department of Computer Science and Electrical Engineering, 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 more 25 years.

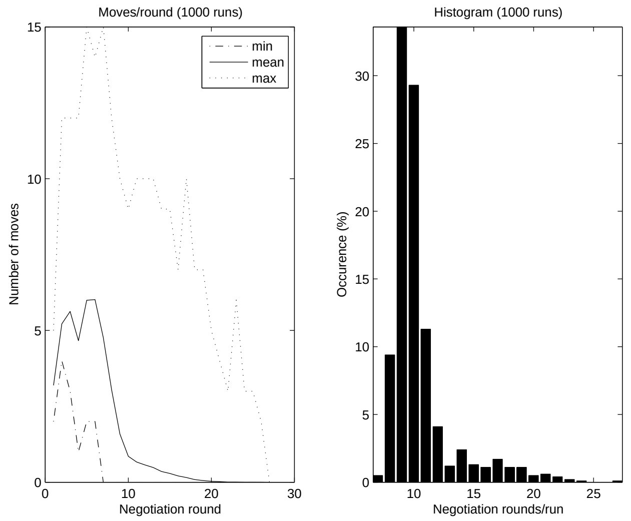
Karl Schoder received his Diplom-Ingenieur in Electrical Engineering from the University of Technology in Vienna in 1997 and his Ph.D. from West Virginia University in 2002. He is a Research Assistant Professor with APERC working on modeling and simulation of electric power systems and FACTS devices. His research interests include modeling, analysis and control of electric transmission and distribution systems.

Shilpa B. Ganesh received a B.S. in Electrical Engineering from Bangalore University, India, in 2000 and an MS in Software Engineering from WVU in 2005. She worked as a software consultant for L&T Komatsu for 3 years, and then as a graduate research assistant at the Advanced Power & Electricity Research Center (APERC) at West Virginia University. She is interested in applying agent technology to control issues of electric power systems.

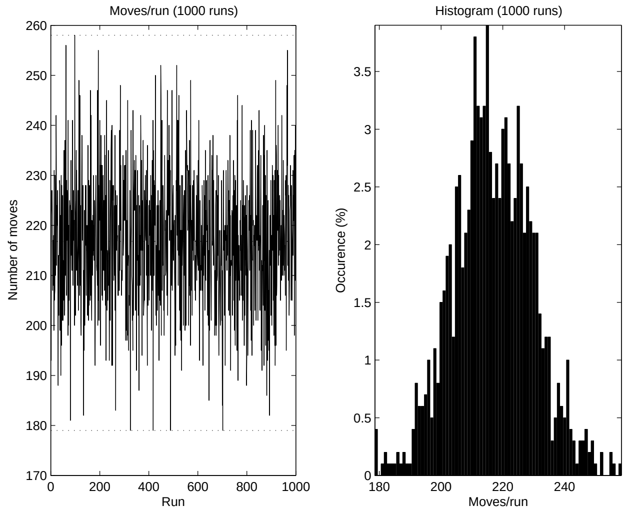
Hong-Jian Lai received his Ph.D. in Mathematics from Wayne State University in 1988. He is a full Professor of the Department of Mathematics at West Virginia University. He has been working on discrete mathematics, algorithms, and optimization for 15 years.



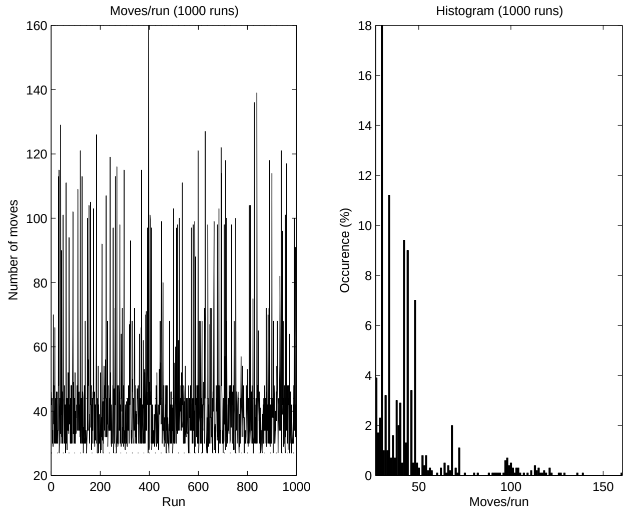
(a) Number of moves made per negotiation round



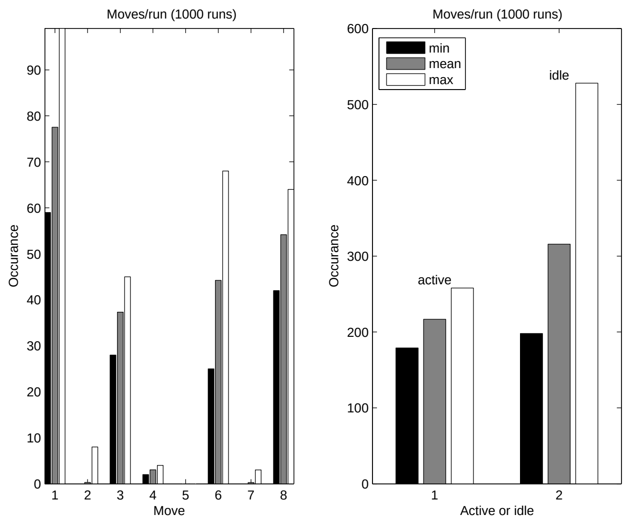
(a) Number of moves made per negotiation round



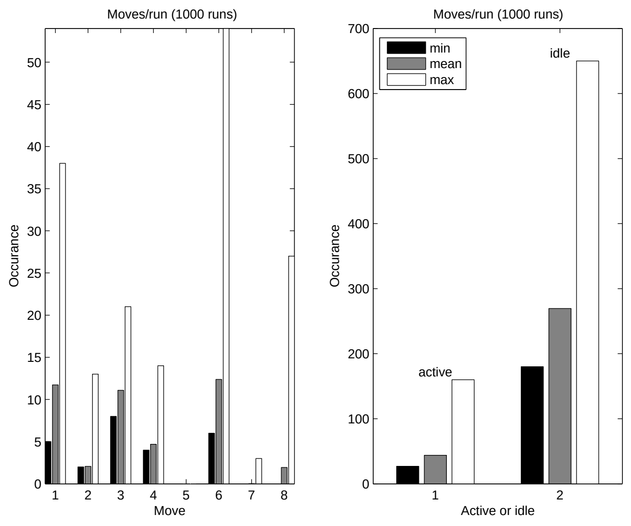
(b) Number of moves made per run



(b) Number of moves made per run



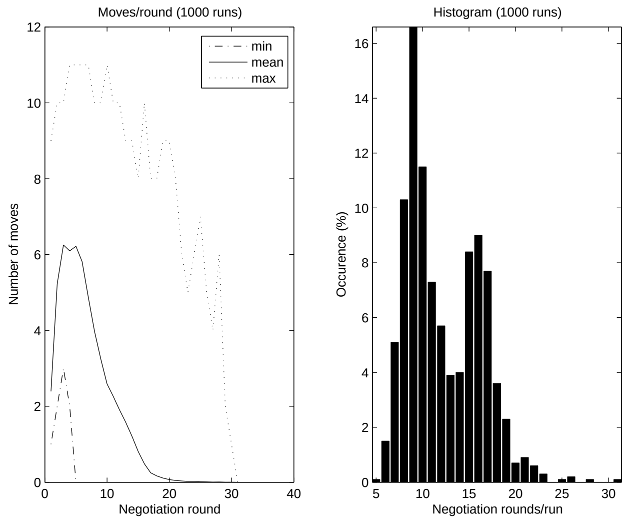
(c) Type of moves made per run



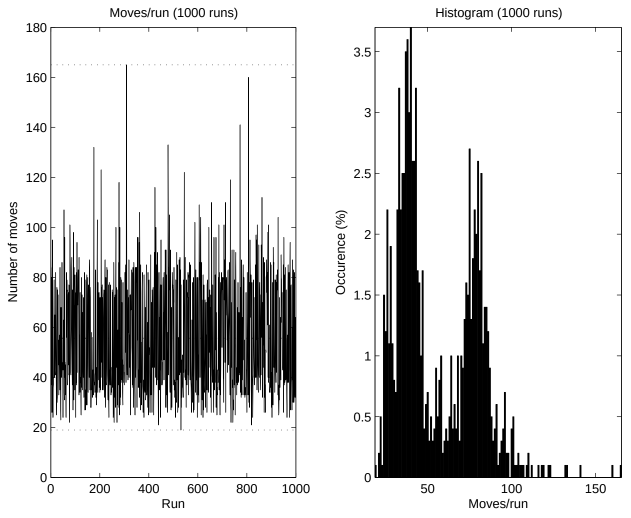
(c) Type of moves made per run

Fig. 4. Agents' activities for the 'startup' scenario

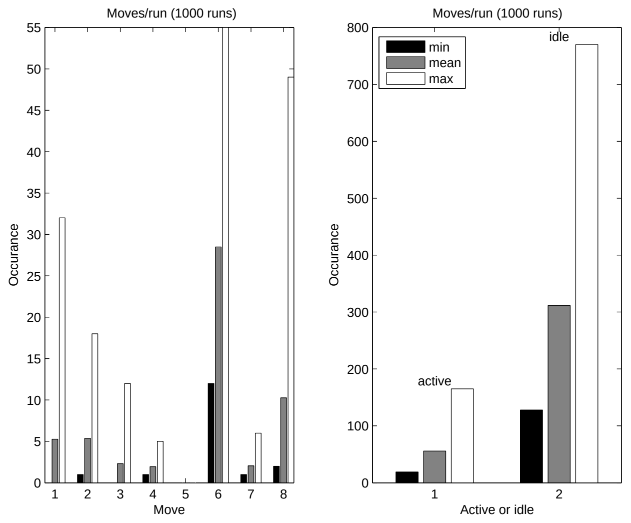
Fig. 5. Agents' activities for the 'loss of link' scenario



(a) Number of moves made per negotiation round

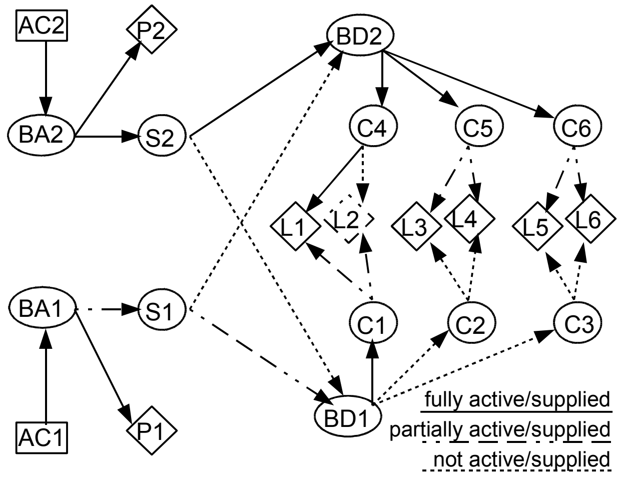


(b) Number of moves made per run

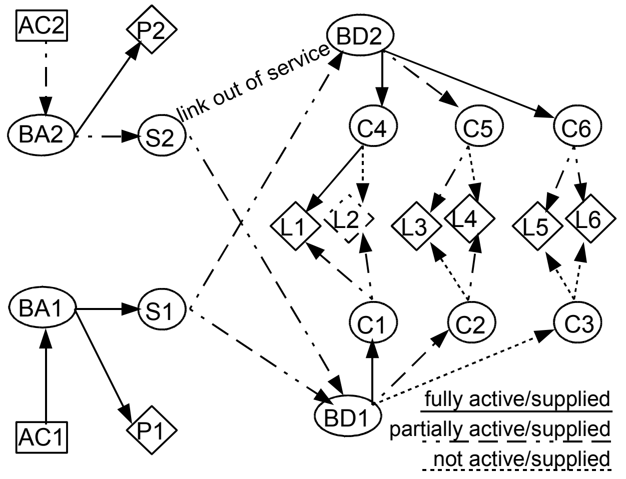


(c) Type of moves made per run

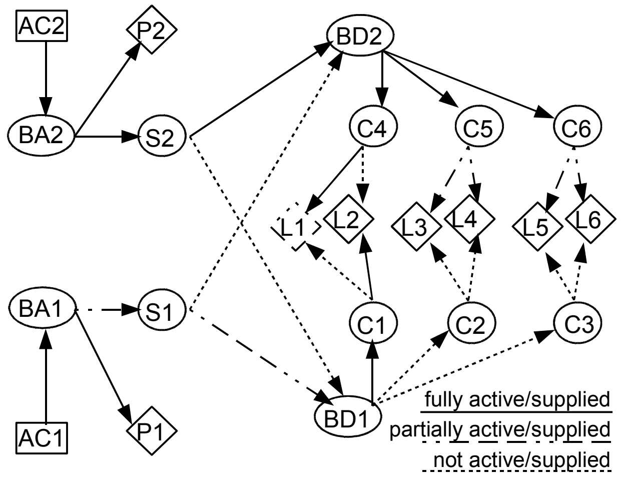
Fig. 6. Agents' activities for the 'priority' scenario



(a) Startup scenario



(b) Loss of link: S2-BD2



(c) Load priority change: p(L2)>p(L1)

Fig. 7. Flow graphs for automatic rerouting scenarios