World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:3, 2009

A Reconfigurable Distributed Multiagent System Optimized for Scalability

Summiya Moheuddin, Afzel Noore, and Muhammad Choudhry

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

Abstract—This paper proposes a novel solution for optimizing the size and communication overhead of a distributed multiagent system without compromising the performance. The proposed approach addresses the challenges of scalability especially when the multiagent system is large. A modified spectral clustering technique is used to partition a large network into logically related clusters. Agents are assigned to monitor dedicated clusters rather than monitor each device or node. The proposed scalable multiagent system is implemented using JADE (Java Agent Development Environment) for a large power system. The performance of the proposed topology-independent decentralized multiagent system and the scalable multiagent system is compared by comprehensively simulating different fault scenarios. The time taken for reconfiguration, the overall computational complexity, and the communication overhead incurred are computed. The results of these simulations show that the proposed scalable multiagent system uses fewer agents efficiently, makes faster decisions to reconfigure when a fault occurs, and incurs significantly less communication overhead.

Keywords—Multiagent system, scalable design, spectral clustering, reconfiguration.

I. Introduction

A Multiagent system (MAS) is a collection of collaborating intelligent and autonomous computational entities called agents [1], perceiving the environment using sensors and acting upon it through actuators [2]. The flexibility and adaptability of multiagent systems make them attractive for several real world applications. Multiagent systems are suitable where scalability is an important requirement, especially where the number of entities to be monitored and controlled is large, processing is performed in a distributed and parallel environment, and there is an exponential growth in the computational complexity. This has been demonstrated by using the agent paradigm to build a biological model simulating polypeptide generation [3]. A combination of cluster computing and multiagent systems has also been used for achieving scalability. This concept is used in a distributed multiagent model for network-based financial computing and economic analysis [4], and for developing scalable software systems on heterogeneous networks [5].

The performance and scalability of a multiagent system is measured using indicators such as number of concurrent agents and associated tasks, organizational pattern of agents, coordination protocols used, and communication and computational overheads [6]. Coordination policies employed by agents are an important indicator of scalability as these are used to determine the total number of messages necessary to converge to a solution [7]. The authors also discuss additional dimensions along which a distributed solution using agent technology needs to scale, including total number of agents, size of data and agent diversity. In [8] the authors analyze the increase in communication load in terms of the number of messages, as the number of agents connected in a number of different topologies is increased. The concept of self building and self organizing MAS to achieve higher scale tolerance by defining scalability in terms of the total processing requirements for agents has been introduced in [9]. MAS that have fixed organizational structures are less scalable than those that adapt to the demands of an application. The issues related to scalability and communication overheads involved in model based teamwork approaches using agents which do not rely on accurate models and further makes use of dynamically evolving and overlapping sub teams are discussed in [10]. Another approach to scalability is to employ a locality model to limit the number of agent interactions thereby making an agent interact only with agents located in a local region [3].

While previous research has broadly focused on achieving scalability by inspecting the organizational patterns of multiagent communities, we investigate achieving scalability by optimally reducing the total number of agents in the multiagent system without compromising the performance. The reduction in the number of agents decreases the resource requirements, simplifies the agent system topology, reduces the communication overhead by limiting the number of possible interactions and minimizes the overall complexity of the system. The proposed scalable multiagent architecture is applied to power system reconfiguration as a case study.

II. MAS applications in electric power systems

The evolution of electric power industry due to recent restructuring and deregulation has led to a major paradigm shift from centralized system towards decentralized solutions. Motivations for this change are discussed in [11]-[13]. Considering power system reconfiguration in particular, centralized solutions based on artificial neural networks, genetic algorithm and expert systems have been proposed [14], [15]. However, such solutions do not adequately address the requirements of modern power systems. Efficient monitoring, control, restoration and reconfiguration of power systems require distributed decentralized control. Multiagent systems have been applied for solving problems such as reconfiguration, restoration, fault identification, diagnosis and power system protection [16]-[19].

Most of the work employing multiagent systems for power system reconfiguration and restoration use an agent to rep-

S. Moheuddin, A. Noore, and M.A. Choudhry are with the Lane Department of Computer Science & Electrical Engineering,West Virginia University, USA.
Manuscript received February 21, 2009

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

resent an electric component such as buses, switches, and circuit breakers etc. [11], [20], [21]. A major problem with such solutions is the limited application especially for large scale power systems with hundreds of components. The size of the multiagent system used in addressing the problem and interactions among agents introduce challenges due to scalability.

Another issue that becomes more and more important as the size of target system increases is that of communication overhead. As discussed previously, communication and coordination policies used in a multiagent system can have serious impact on scalability. The issue of increased communication overhead has been discussed in a multiagent based algorithm for reconfiguration of ring structured shipboard power systems in [22], [23]. Similar concept for reconfiguration has been proposed for mesh structure power systems in [24]. Message passing is used for the identification of a ring structure to avoid accumulation of redundant information in the system. As the size of the power system increases, the communication overhead also increases. Huang et al. [25] introduce a purely decentralized approach to address this issue. The exponential increase in message or communication overhead with the number of loads in a power system and the resulting delay in decision making has been discussed in [26]. Their solution uses mobile agents for local data utilization and determining the impact of power system loads on power quality. Although such solutions show better performance than a centralized approach, further improvement can be achieved by optimizing the size of agent communities and their organizational patterns.

Recently, distributed intelligence has been applied to switch controls using the concept of teams and coaches [27], [28]. A team represents line segments bounded by switches, while coaches have the job to monitor and coordinate their teams and perform restoration by communicating with neighboring teams. This approach considers the availability of multiple sources and coaches try to identify alternate sources that can supply the load in their team. The coaches move to team members to determine the state of team line segment. This design decision makes agents more susceptible to faults. Also, every node has to be configured for starting agents by providing hardware and software support. On detecting faults, team members communicate this information to other team members and to the coach.

There is also the need for a common place where information and measurements are collected and appropriately coordinated, even in approaches that are perfectly autonomous and distributed. This is especially useful in cases where one out of several possible options is to be selected based on some criteria. For example, in [20] junction agents act as coordinators/controllers for negotiation between different bus agents linked to the transmission line. A common coordinating entity has also been used in [29] where multiagent technology has been applied to the dynamic reconfiguration of power systems by considering time varying loads. In [25] the use of a root agent to compute the net power of the system by collecting information from all its children is discussed. This root agent also acts as a central location where information is accumulated before any decisions are made.

A major challenge of agent architectures for large scale power systems is scalability. Most existing architectures, associate an agent with an electronic component in the system. The idea is to have agents with local information communicate with each other to reach a solution. However, the amount of communication, coordination and other resource requirements for such solutions will soon outweigh the anticipated benefits of decentralization. So, no matter how promising the suggested multiagent system is, its size will always pose a problem. This triggers the need to find a trade-off between decentralization and size of the multiagent system. Efficient solutions would place a bound on the number of agents in the system.

In this research, we have developed clustering techniques to identify logical power system partitions such that agents can monitor these dedicated clusters or partitions rather than employ an agent to monitor each node or device. This reduces the resources utilized and the complexity of the system. Our proposed approach significantly reduces the number of agents in the system and the communication overhead while minimizing the system complexity and computational overhead. Furthermore, in the event of a fault, the affected nodes are quickly identified for reconfiguration.

Section III proposes a decentralized MAS architecture. The agents in this decentralized MAS use only local information and the MAS is topology-independent. Section IV describes the proposed partitioning of large power system using spectral clustering based on the properties of electrical distances. Section V introduces the proposed scalable MAS architecture which is applied to the partitioned power system. We use the decentralized MAS as a baseline to compare the performance of the proposed scalable MAS and the time taken to reconfigure in the event a fault is detected. The implementation details of the proposed multiagent system are presented in Section VI. Simulation results obtained for the decentralized MAS and the scalable MAS are analyzed in Section VII.

III. Proposed topology-independent decentralized multiagent system architecture

The proposed topology-independent MAS uses three types of agents and their functions are briefly described.

Bus Agent (BA) represents buses in the power system. Agents representing neighboring buses are defined as neighbors in the corresponding multiagent system. As a completely decentralized approach, only local information is used by these agents. The agents have no topology information and only neighboring agents in the system are allowed to communicate with each other.

Processor Agent (PA) represents a common place in the multiagent system where information from other agents is accumulated. PA is the common coordinating entity in the system. It is used to collect the dynamic operational profile of the system and identify nodes that are affected in the system in the event of a fault.

Switch Agent (SA) monitors and controls switches in the power system and performs the task of turning a switch on or off.

The proposed topology-independent decentralized MAS was applied to the *Circuit of the Future* (COF) system, developed

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

by Southern California Edison (SCE) [30]. The CoF has a single substation with three main feeders. These feeders are also connected for flexible re-routing of power flow, through seven switches which are normally open. There are 14 loads in the circuit with a total real power demand of 24 MW and reactive power load of 12.96 MVar. There are a total of 66 buses and 7 switches in the system requiring 66 *bus agents* and 7 *switch agents*.

A. Identification and reconfiguration of nodes affected by fault

As soon as a *bus agent* detects a loss of power, it starts communicating with its neighboring agents about the problem. A message is sent to each neighbor, to determine the status of power availability. If the neighboring agents also do not have power, they forward the message further to their neighbors. As the messages are forwarded, agents append their identifiers to the message. When the message reaches an agent whose corresponding bus has power or reaches an agent whose identifier is already in the message content, the forwarding stops. The flow of message also stops at the *terminal nodes* (TN). We define *terminal nodes* as buses corresponding to agents which have no neighbors except the sender of the message. When a message reaches a *terminal node*, the corresponding agent appends its identifier to the message content and forwards it to the *processor agent* (PA). The information about different paths a message took generates the knowledge of all nodes or buses involved in a fault. Therefore, every *bus agent* where the message flow stops, forwards the final message it receives to the *processor agent*. The *processor agent* then processes the message received and compiles a list of all the agents affected by the fault.

While disseminating information about the fault, if a bus has a neighboring switch, the corresponding *bus agent* sends a message to the *bus agent* at the other end of the switch to determine if it has power. If the other *bus agent* informs that its corresponding bus has power, by turning the switch on, the de-energized area can be supplied power. A message is sent to the *processor agent* suggesting the switch position as a potential choice for reconfiguration. However, there can be situations where buses at both ends of the switch are without power. Such a switch is ruled out as a choice for reconfiguration. The *processor agent* compiles a list of all the suggested switch positions and then selects a switch or a possible combination for reconfiguration. *Switch agents* corresponding to these selected switches control the final switching operation for reconfiguration. The coordination of different tasks by all the agents involved in fault identification and reconfiguration is described in Algorithm I.

Fig.1 is a graphical representation of the *Circuit of the Future* (CoF). Using this circuit the algorithm is described for various fault scenarios and the interaction among agents to perform reconfiguration is discussed. With Bus 1 as the source node, suppose there is a fault between Bus 5 and Bus 17 as indicated by Fault 1 in the figure. *Bus agent* at Bus 5 i.e. BA5 senses that it has no power and forwards the message to its neighbors BA17, BA6 and BA14. Since BA17 has power, the message flow stops there. The other neighbors of BA5 keep forwarding the message by adding their ID. When the message reaches BA6 it has two possible paths.

One is to BA10 and the other is to BA7. The message flow continues as normal along the path through BA7. On reaching BA10, since it is a *terminal node* the message flow stops. At this point, BA10 forwards the message to the PA. The same steps take place at Buses 16, 15, 13, 14, since they are all *terminal nodes*. Also, BA10 suggests to PA the neighboring switch SW6 as a possible option for reconfiguration after consulting BA63. BA15, BA5, and BA8 also recommend their neighboring switches SW1, SW2 and SW4, SW7 respectively to PA. However, SW5 between Buses 13 and 14 does not make the list of possible switches for reconfiguration as buses at both end of the switch are de-energized. After *bus agents* have finished communicating, the *processor agent* will have acquired knowledge of the nodes affected by the fault. The *processor agent* then uses this information and suggested

Algorithm I

(Function and coordination of agents after a fault is detected between Bus i and Bus j)

Bus Agent

```
BAj communicates with neighbors to disseminate fault
information
An agent BAx receives a message and performs the
following tasks:
IF BAx has power and a neighboring switch THEN
Inform sender that BAx is energized
ELSE
    IF BAx has neighboring switch THEN
    Check if bus at other end of switch is energized
         IF bus is energized THEN
         Suggest switch position to PA
         ENDIF
     ENDIF
     IF BAx is at terminal node THEN
     Send an intermediate list of agents affected by
     a fault to PA
     ELSE
     Add self to the list of agents affected by the
     fault
     FOR all elements in the list of neighbors
     Send the list generated in previous step
     ENDFOR
ENDIF
```

Processor Agent

```
Determines all agents affected by the fault
Determines all potential switches for reconfigura-
tion
Selects a switch for reconfiguration
```

Switch Agent

```
Performs the final task of changing the switch
status
```

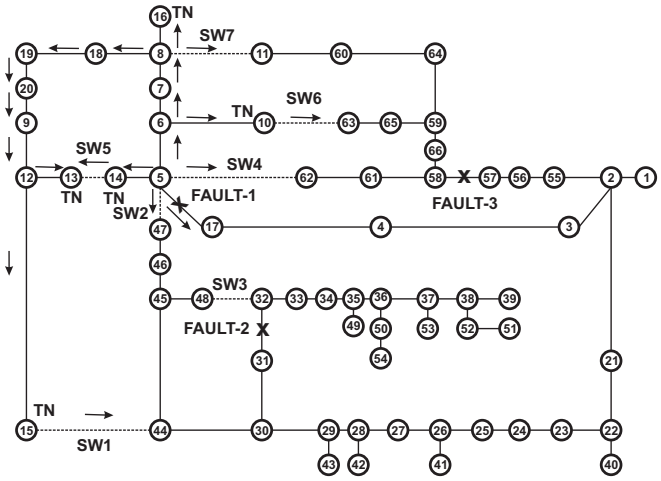

Fig. 1. Graphical representation of the *Circuit of the Future* and example fault scenarios

switch positions to direct the *switch agents* to perform the switching operation for reconfiguration.

All possible single fault scenarios for the test system are simulated using the proposed topology-independent decentralized multiagent system. Some example scenarios and results are listed in Table I. Although the same number of nodes is involved in Fault 1 and Fault 2 in Table I, the number of messages passed is different. The reason is that the communication involved in any fault scenario has two main components. The first component pertains to the messages for propagating the information about the fault. This happens between the *bus agents*. The second component is the communication between the *bus agents* and the *processor agent*. This depends on the number of *terminal nodes* affected by a fault and the number of nodes having neighboring switches. Therefore, although the first communication component in both the faults is the same, it is the second component that makes the difference.

An analysis of the proposed topology-independent decentralized MAS architecture and the agent interactions shows that the size of the multiagent system is very large. The agents in the system mostly perform simple message forwarding tasks while consuming resources. Also, the large communication overhead is due to communication between agents when a fault is detected and to determine the affected nodes. Depending upon the number of nodes involved in a fault, this overhead can cause significant delays in decision making. After the affected nodes have been identified, a switching position is selected for reconfiguration. While the proposed approach works well for power systems of smaller size, for larger power systems we need a scalable solution to perform effective reconfiguration when a fault occurs. The design of a scalable MAS architecture in presented in Section V.

IV. Spectral Clustering Techniques for Power System Partitioning

In this section we propose a spectral clustering algorithm to logically partition the power network into clusters of connected buses. In the algorithm we select a clustering parameter that represents the notion of electric distances. To tune the clustering algorithm for partitioning a power system and to give importance to electrical properties of the system, the bus impedance matrix is used to acquire the necessary information for clustering. Agents are then assigned to each cluster or partition. This modified approach scales well as the size of the power system increases by reducing both the total number of agents in the system and the communication overhead. However, to gain these benefits, the agents in the clusters acquire additional knowledge about the topology in their cluster.

By partitioning a power system network into strongly connected components significant performance gains are obtained. It has been shown that many control actions and small disturbances impact only a small portion of the system [31]. Also, power system applications that require load-flow calculations and contingency analysis take more time as the size of the matrices increases. So dimensionality reduction by focusing only on the most affected portion of the network can be highly efficient. We first represent the power system as a weighted graph $G(V, E)$ where every bus in the power system is defined as a vertex/node. The edges of the graph represent the connection between different buses in the power system and the edge weights are based on the electrical distances between different buses in the power system. It is important to take into consideration the electrical properties of the clusters, if a partitioning algorithm is to be applied to a power system graph. We have chosen the electrical distances between different buses as the clustering parameter. These distances are obtained from the bus impedance matrix, *Zbus*. Each element in the symmetric *Zbus* matrix indicates the electrical closeness of two buses in a power system. The closer the buses in the power system, the lower the value of the corresponding *Zbus* elements. In our clustering algorithm, the final edge weights are obtained by inverting the absolute values of the *Zbus* matrix so that the edge weights are higher when the buses are closer and vice versa.

A. Spectral clustering for partitioning a power system

Clustering is the process of identifying the underlying structure in data and determining groups of similar behavior [32]. Spectral clustering algorithm is based on the spectral graph theory. The spectrum of a matrix representing a graph is analyzed by computing the eigen vectors of a graph, and ordering by the magnitude of their corresponding eigen values. Spectral clustering algorithms usually outperform traditional clustering algorithms [33]. The clustering algorithm used in the proposed architecture is based on the work done by Zelnik-Manor and Perona [34] and Ng et al. [35]. Their work introduces local scales to improve clustering performance and the automatic determination of number of groups that naturally exist in the data by exploiting the eigen vector structure. We briefly present a general outline of the spectral clustering algorithm.

First, a matrix representation of the large scale structure is constructed. Eigen values and eigen vectors of the matrix are calculated. These eigen values and vectors provide global

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

TABLE I
SIMULATION RESULTS OF SELECTED FAULT SCENARIOS IN THE TOPOLOGY-INDEPENDENT DECENTRALIZED MAS

Fault No.	Source Node	Destination Node	Nodes Affected by a Fault	No. of Nodes Affected	Messages Passed	Time Taken (ms)	Suggested Switch Positions for Reconfiguration
1	17	5	5, 6, 7, 8, 10, 16, 18, 19, 20, 9, 12, 13, 14, 15	14	49	40.7	SW-4, SW-2, SW-6, SW-7, or SW-1
2	31	32	32, 33, 34, 35, 36, 49, 50, 54, 37, 53, 38, 52, 39, 51	14	31	31.5	SW-3
3	57	58	65, 11, 60, 64, 59, 66, 58, 61, 63, 62	10	29	46.7	SW-4, SW-6, or SW-7

information about the structure of the matrix and its connectivity. Also, a mapping of data points to a lower dimensional representation is performed based on one or more eigen vectors. Finally, the data points are assigned to different clusters. We tailor the spectral clustering algorithm for partitioning a power system. The modified algorithm is used on a WSCC-9 bus system [36]. Let S represent the set of buses in the power system. Then,

$$N = |S| \quad (1)$$

$$A = [a_{ij}] \quad \text{is the N x N adjacency matrix} \quad (2)$$

$$Z = [Zbus_{ij}] \quad (3)$$

$$a_{ij} = \frac{1}{abs(z_{ij})} \quad (4)$$

After obtaining the adjacency matrix, the N x N diagonal degree matrix is obtained using

$$D(i,j) = \sum_{j=1}^{N} A(i,j) \quad (5)$$

The spectral clustering algorithm uses the normalized Laplacian matrix defined as,

$$\hat{\mathrm{A}} = D^{-(\frac{1}{2})} A D^{-(\frac{1}{2})} \quad (6)$$

$\hat{\mathrm{A}}$ contains all the information necessary to perform spectral clustering. The eigen vectors and eigen values of this matrix are used to determine the specified number of groups K, in the original power system. The K largest eigen vectors of $\hat{\mathrm{A}}$ are then used to obtain an N x K matrix, V. The rows of V are renormalized to have unit length generating an N x K matrix Y. Each row of Y acts as a point in the K dimensional space. K-means algorithm is then applied on this matrix for clustering.

The buses in the WSCC-9 bus power system are transformed into 9 vertices of a graph. The edges in the graph represent the connectivity between the buses. The WSCC-9 bus system and the corresponding graph are shown in Fig. 2. We first construct the adjacency matrix A from the weighted graph. There are a total of nine buses in the power system. This results in a 9 x 9 adjacency matrix representing the adjacency between the buses of the power system.

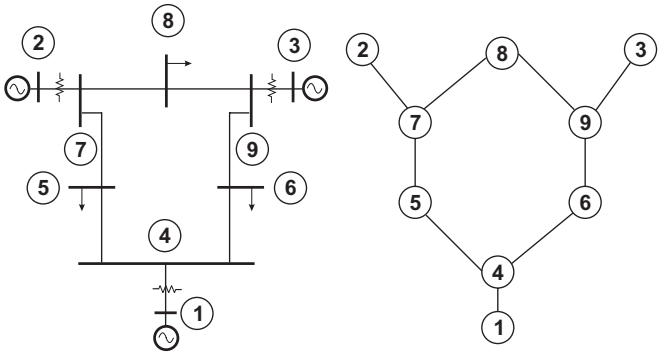

Fig. 2. WSCC-9 bus system and its corresponding graph

In our proposed algorithm we have modified the adjacency matrix to represent the notion of electric distances. A bus in a power system can be electrically closer to another, although not directly connected. To give importance to such electrical properties and to fully capture the information provided by the *Zbus*, the modified adjacency matrix we used has a different structure. The resulting 9 x 9 adjacency matrix A for the WSCC-9 bus system is shown. It can be observed that the diagonal entries in the matrix A are the largest in each row since a bus is electrically closest to itself. We have experimentally verified that this choice of adjacency matrix based on electrical properties of the system generated better results. The 9 x 9 diagonal degree matrix is obtained from the adjacency matrix using Equation (5). Finally, using both the adjacency matrix and the degree matrix, we obtain the 9 x 9 Laplacian matrix $\hat{\mathrm{A}}$ using Equation (6). The matrix $\hat{\mathrm{A}}$ for the WSCC-9 bus system is shown. The eigen vectors and eigen values of the Laplacian matrix are calculated. From these eigen vectors the 9 x 3 matrix V is constructed, where v_1, v_2, and v_3 are the three largest eigen vectors of $\hat{\mathrm{A}}$. The structure of the eigen vectors of the Laplacian matrix encodes the structure of the resulting clusters.

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

$$A = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \\ \mathbf{7} \\ \mathbf{8} \\ \mathbf{9} \end{array} \begin{array}{c} \begin{array}{ccccccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} & \mathbf{9} \end{array} \\ \begin{pmatrix} 1.6039 & 1.3134 & 1.3082 & 1.4702 & 1.3930 & 1.3873 & 1.3134 & 1.3014 & 1.3082 \\ 1.3134 & 1.6144 & 1.3362 & 1.3134 & 1.3443 & 1.3010 & 1.4664 & 1.3986 & 1.3362 \\ 1.3082 & 1.3362 & 1.6001 & 1.3082 & 1.3001 & 1.3378 & 1.3362 & 1.3758 & 1.4630 \\ 1.4702 & 1.3134 & 1.3082 & 1.4702 & 1.3930 & 1.3873 & 1.3134 & 1.3014 & 1.3082 \\ 1.3930 & 1.3443 & 1.3001 & 1.3930 & 1.4697 & 1.3380 & 1.3443 & 1.3156 & 1.3001 \\ 1.3873 & 1.3010 & 1.3378 & 1.3873 & 1.3380 & 1.4682 & 1.3010 & 1.3062 & 1.3378 \\ 1.3134 & 1.4664 & 1.3362 & 1.3134 & 1.3443 & 1.3010 & 1.4664 & 1.3986 & 1.3362 \\ 1.3014 & 1.3986 & 1.3758 & 1.3014 & 1.3156 & 1.3062 & 1.3986 & 1.4639 & 1.3758 \\ 1.3082 & 1.3362 & 1.4630 & 1.3082 & 1.3001 & 1.3378 & 1.3362 & 1.3758 & 1.4630 \end{pmatrix} \end{array}$$

$$\hat{A} = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \\ \mathbf{7} \\ \mathbf{8} \\ \mathbf{9} \end{array} \begin{array}{c} \begin{array}{ccccccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} & \mathbf{9} \end{array} \\ \begin{pmatrix} 0 & 0.1293 & 0.1295 & 0.1162 & 0.1204 & 0.1205 & 0.1293 & 0.1287 & 0.1295 \\ 0.1293 & 0 & 0.1270 & 0.1293 & 0.1245 & 0.1281 & 0.1165 & 0.1203 & 0.1270 \\ 0.1295 & 0.1270 & 0 & 0.1295 & 0.1281 & 0.1245 & 0.1270 & 0.1219 & 0.1162 \\ 0.1162 & 0.1293 & 0.1295 & 0 & 0.1204 & 0.1205 & 0.1293 & 0.1287 & 0.1295 \\ 0.1204 & 0.1245 & 0.1281 & 0.1204 & 0 & 0.1226 & 0.1245 & 0.1253 & 0.1281 \\ 0.1205 & 0.1281 & 0.1245 & 0.1205 & 0.1226 & 0 & 0.1281 & 0.1259 & 0.1245 \\ 0.1293 & 0.1165 & 0.1270 & 0.1293 & 0.1245 & 0.1281 & 0 & 0.1203 & 0.1270 \\ 0.1287 & 0.1203 & 0.1219 & 0.1287 & 0.1253 & 0.1259 & 0.1203 & 0 & 0.1219 \\ 0.1295 & 0.1270 & 0.1162 & 0.1295 & 0.1281 & 0.1245 & 0.1270 & 0.1219 & 0 \end{pmatrix} \end{array}$$

$$V = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \\ \mathbf{7} \\ \mathbf{8} \\ \mathbf{9} \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{pmatrix} -0.3344 & -0.4768 & 0.0161 \\ -0.334 & 0.2936 & 0.4534 \\ -0.3344 & 0.2724 & -0.4998 \\ -0.3344 & -0.4768 & 0.0161 \\ -0.3316 & -0.2469 & 0.2048 \\ -0.3318 & -0.2499 & -0.2063 \\ -0.334 & 0.2936 & 0.4534 \\ -0.3312 & 0.3181 & 0.0641 \\ -0.3344 & 0.2724 & -0.4998 \end{pmatrix} \end{array}$$

A closer look at the second column corresponding to v_2 shows the structure of the resulting clusters or partitions. Buses that will be placed in the same cluster have very similar values, differing by a small fraction. All buses having negative eigen values in eigen vector v_2 will be grouped in the same cluster. Cluster 1 includes Buses 1, 4, 5, and 6. Similarly Cluster 2 includes Buses 3 and 9, and Cluster 3 includes Buses 2, 7, and 8.

The proposed spectral clustering algorithm was extended to the Southern California Edison's *Circuit of the Future* (CoF), which is a larger system compared to the WSCC-9 bus system. As shown in Fig. 1 there are a total of 66 buses in this system [30]. A 66 x 66 Laplacian matrix is obtained. As before, the eigen values are used to group the 66 buses into 5 clusters. Table II summarizes the details of the five clusters in the CoF after applying the proposed spectral clustering algorithm.

TABLE II
RESULTS OBTAINED AFTER APPLYING SPECTRAL CLUSTERING ON CIRCUIT OF THE FUTURE

Cluster No.	Total Bus Count	Buses in the Cluster
Cluster 1	22	1, 2, 3, 55, 56, 57, 58, 61, 62, 66, 59, 65, 63, 64, 60, 11, 21, 22, 40, 23, 24, 25
Cluster 2	16	4, 17, 5, 6, 14, 7, 10, 8, 16, 18, 19, 20, 9, 12, 13, 15
Cluster 3	13	47, 46, 45, 44, 48, 30, 29, 28, 43, 42, 27, 26, 41
Cluster 4	6	31, 32, 33, 34, 35, 49
Cluster 5	9	36, 50, 54, 37, 53, 38, 52, 39, 51

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

V. Proposed Scalable Multiagent System Architecture

In this section we describe a scalable multiagent architecture using fewer agents. The MAS architecture uses two types of agents.

Cluster Agent is assigned to each cluster or partition instead of assigning an agent to each bus as proposed in the decentralized topology-independent MAS. This results in a smaller agent system. We designate these agents as *cluster agents* or CAs. Fig. 3 shows the 5 *cluster agents* CA1-CA5 and 7 reconfiguration switches, SW1-SW7. These *cluster agents* can communicate with each other and work in coordination to perform different functions. By dividing the large system into logical clusters and assigning these to different agents we reduce the size and complexity of the problem for each agent, thereby making the load flow calculations, monitoring, reconfiguration, or restoration efficient and manageable.

Switch Agent turns a reconfiguration switch on or off based on the chosen path and given constraints.

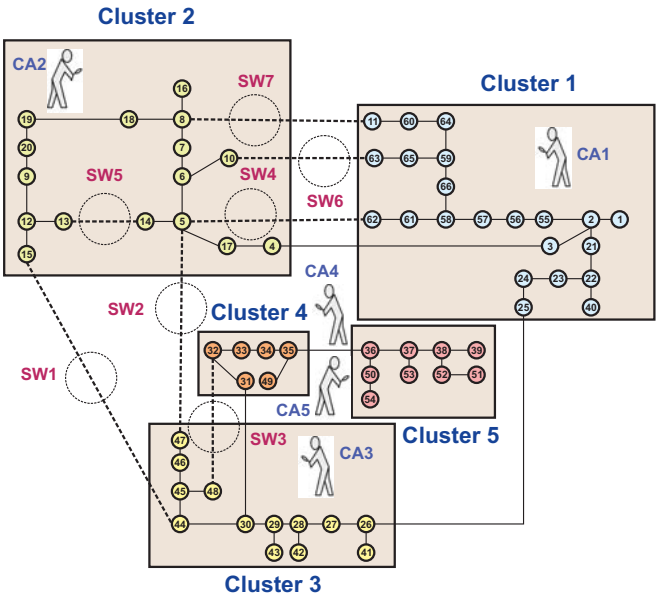

Fig. 3. Assigning cluster agent CA_i to each cluster obtained in the *Circuit of the Future*

A. Functionality of the proposed scalable multiagent system

Each CA has knowledge of the topology of the nodes or buses in the cluster it is monitoring. It is provided in the form of a matrix which contains the weighted adjacency information of only the buses in the cluster associated with an agent. For example, there are 16 buses in the cluster associated with CA2, and so a 16 x 16 symmetric matrix is provided to the associated CA. Thus each agent has a lower dimensional matrix compared to the actual system size. The overall algorithm describing the interaction between agents in the proposed scalable MAS is described in Algorithm II.

The edge weights forming entries of the weighted adjacency matrix can represent different constraints in the system. Once

Algorithm II

Step 1: Obtain a weighted graph G(V,E) from the power system

Step 2: Apply spectral clustering algorithm to obtain K partitions

Step 3: Assign agents to clusters obtained Interaction of cluster agent CA_j with a faulty cluster agent CA_i

Step 4: CA_j sends a list of potential source nodes and associated costs to CA_i

For a cluster agent with fault

Step 4a: Dynamically update model of cluster topology

Step 5: CA_i communicates with its neighboring *cluster agents* for finding a solution

Step 6: CA_i determines nodes affected by the fault

Step 7: CA_i performs analysis of each alternate source node Si in parallel

Step 8: CA_i evaluates potential reconfiguration solutions from neighbors

Step 9: CA_i determines a feasible best solution based on constraints

Step 10: CA_i communicates with selected *switch agent* to perform the final reconfiguration

a fault is detected, the CAs can identify possible solutions and an optimal solution is selected based on these constraints. For example, edge weights can represent the power flow between each node. These edge weights can also be assigned the values of voltage drop between nodes. In this manner we can identify paths which have minimum voltage drops. Also, priorities can be encoded into these edge weights. For example, if certain paths in the power system are known to be more reliable than others, this information can be encoded in the edge weights such that their priority is reflected in the decision making process. A cost function is used to identify an optimal solution for reconfiguration, depending on system constraints. We define the cost function as,

$$Cost = \sum_{i=1}^{K} C(i)W(i) \tag{7}$$

where W(i) represents the weight of each edge in the path, K is the length of the path and $C(i)$ is a Cost Adjustment Factor, which models different constraints. For example, it can be used to give importance to load priorities when both a high priority load and a low priority load need power at the same time. $C(i)$ can be adjusted to minimize the cost of the path to the high priority load making sure it is served before any other load in the system, or the $C(i)$ can be adjusted to select a more reliable path by reducing the overall cost of the path. Since our aim for this case study is the demonstration of effectiveness of *cluster agents* interaction, we consider simple fault scenarios resulting in loss of line. A fault is modeled as a loss of line

between two buses in the power system where certain nodes in the power system are unsupplied.

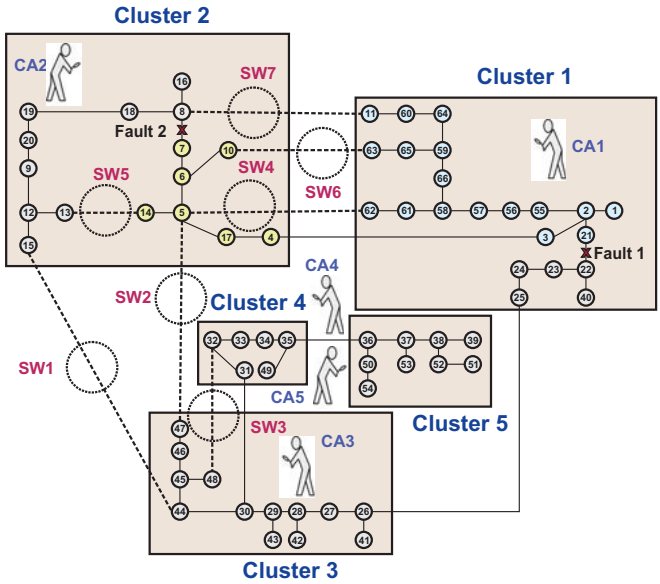

Fig. 4. Example fault scenarios in the *Circuit of the Future*

B. Identification and reconfiguration of nodes affected by fault

When a fault is first detected in a particular cluster, the *cluster agent* associated with that cluster updates its model of cluster topology. It then determines the nodes affected by the fault. The agent determines the minimal cost of paths between its current source node and all other nodes in its graph structure. The agent also finds the parent nodes of all the nodes involved in the shortest path from the source node. In this manner an agent acquires knowledge of the paths and the associated costs for all nodes in its cluster. After application of this algorithm, all the nodes for which the cost of the path is greater than a maximum value are identified as those affected by the fault.

Certain faults can propagate from one cluster to neighboring clusters. Under such circumstances, the *cluster agent* associated with the cluster where the fault actually occurs first identifies a list of neighbors which can be potentially affected by the current fault in the system. The *cluster agent* then informs these neighbors about the problem. The neighboring *cluster agents* also update their model of the cluster topology and pass this information to its neighbors which can be affected by the newly introduced fault in their cluster. All affected *cluster agents* get involved in the process of reconfiguration in parallel. Fig. 4 shows such a scenario where a fault in Cluster 1 propagates to Cluster 3, Cluster 4, and Cluster 5. The affected buses in each cluster are highlighted in gray. The four *cluster agents* affected by the fault between Bus 21 and Bus 22 in Cluster 1, work together to reach a decision for reconfiguration.

C. Identification of alternate source nodes

For a cluster, a source node is defined as any node which is connected to a node in some other cluster. These are called source nodes because they have the potential to provide power to the de-energized area by switching on the nearest reconfiguration switch. The agent acquires the position of source nodes at start up. Some of these source nodes will be at a location and will have connectivity, such that they can never provide power to the de-energized buses. For a particular source node, these will be the buses un-reachable from it based on the new network connectivity information after the fault. So the agent tries to rule out all such source nodes. The *cluster agent* for the affected cluster also considers the option of internal reconfiguration if it has switches inside its cluster. Meanwhile, agents in the neighboring clusters also perform their computations for finding the best route to redirect the power to the faulty area. It is important to note that since each agent is a separate thread, all these computations are performed in parallel. The neighboring agents determine the cost from their currently active source node to the nodes which can potentially provide power to the affected cluster. This information is shared with the CA in the affected cluster. Since time is critical in such situations, to reach a decision quickly the CA has already determined the nodes which cannot act as source node under the current fault and rules them out. For the rest of the nodes, the *cluster agent* chooses the source node with the minimal cost to the affected nodes. In this manner the agent with the help of its neighbors determines the optimal path for reconfiguration, after a fault is detected. The switching action is then performed to energize the unsupplied buses through the newly configured path.

As an example, let us assume there is a fault between Bus 7 and Bus 8, in the cluster assigned to CA2 as shown in Fig. 4. When the fault is detected, CA2 communicates with its neighbors CA1 and CA3, to initiate the task of identifying possible reconfigurable paths and computing their associated costs. CA2 determines the nodes affected by the fault. The affected nodes are highlighted in gray in Fig. 4. The neighboring agents perform their calculations to suggest alternate routes and associated costs to CA2. CA2 determines the feasibility of different source nodes. Notice in this case that switching on SW2, SW4, or SW6 cannot solve the problem since they will not be able to supply power to the affected nodes. So CA2 rules these out. The potential alternate source nodes are identified as Node 11 in Cluster 1, and Node 44 in Cluster 3. The third option involves internal switching within Cluster 2 between Nodes 13 and 14. Out of these options the agent selects the one with the minimum cost. We simulated all possible single faults for SCEs *Circuit of the Future*. Some example scenarios of faults in each cluster of the scalable multiagent architecture are analyzed. The number of messages passed, possible switches needed for reconfiguration and the associated cost, and the time taken to arrive at a solution are summarized in Table III.

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

TABLE III
SIMULATION RESULTS OF SELECTED FAULT SCENARIOS IN THE PROPOSED SCALABLE MAS

Cluster No.	Source Node	Destination Node	Clusters Affected by a Fault	Alternate Source Nodes	Switch for Reconfiguration	Cost	Messages Passed	Time (ms)
C1	2	55	C1	8 10 5	SW7 SW6 SW4	5 4 2	4	24.9
C1	1	2	C1, C2, C3, C4, C5	None	NA	NA	20	35.8
C2	7	8	C2	14 11 44	SW5 SW7 SW1	3 10 5	4	17.1
C3	27	28	C3, C4, C5	15 5	SW1 SW2	11 2	12	36.1
C4	31	32	C4, C5	48	SW3	7	5	32.9
C5	36	50	C5	None	NA	NA	2	14.2
Between C1 and C2	3	4	C2	62 11 63 47 44	SW4 SW7 SW6 SW2 SW1	7 10 9 8 5	4	23.5

VI. IMPLEMENTATION DETAILS OF THE PROPOSED MULTIAGENT SYSTEM

The proposed multiagent system is implemented using JADE (Java Agent Development Environment). The JADE framework simplifies the development and run-time execution of multiagent applications through a FIPA (Foundation for Intelligent Physical Agents) compliant middle-ware. It allows distribution of the agent platform across multiple machines and allows controlling the configuration through a remote GUI. A homogeneous set of APIs (Application Programming Interfaces) is provided that do not depend on the underlying network or the Java language version. A simple set of APIs hide the complexity of the middleware. The communication infrastructure used by agents in the proposed MAS is provided by JADE. JADE follows FIPA standards which enable agents written in different languages and running on different platforms to communicate with each other. JADE messages adhere to ACL (Agent Communication Language) standards. An ACL message has several attributes such as *performative*, *receiver*, *sender*, and *content*. *Performative* defines the type of the message and agents in the implemented MAS take different actions depending upon the message type. For example, all *cluster agents* suggesting alternate sources to a faulty *cluster agent* set the *performative* of the messages sent to *ACLMessage.PROPOSE*. The faulty *cluster agent* on receiving this type of message knows that the content contains proposals sent by neighbors for reconfiguration [37].

A snapshot of agent communication in JADE during a fault scenario is shown in Fig. 5. The agent communication was captured using another feature provided by JADE i.e. the *sniffer agent*. A *sniffer agent* can intercept ACL messages during agent communication and displays them graphically using a notation similar to UML sequence diagrams. It is also a good debugging tool for agent societies as it helps in observing the message exchange between agents.

Another, important feature offered by JADE which has been implemented in the proposed MAS is the concept of *behaviours*. It is common practice in agent programming to have several concurrent active tasks within an agent. One approach for implementing these concurrent activities is to use Java thread programming. However, since Java threads are not designed for large scale parallelism this approach proves to be inefficient. JADE *behaviours* offer an efficient alternative for implementing concurrent tasks within an agent. The proposed MAS uses this feature of JADE as agents are concurrently involved in many different tasks. Examples of these tasks include communication with neighbors, identification of nodes affected by a fault, identification of alternate sources etc. Also, each agent has several *behaviours* running in parallel. For example each of the CAs has a cyclic *behaviour*, which executes continuously for the lifetime of the agent and receives messages indicating a fault in the respective cluster. These messages are sent by a dedicated agent in the system, which receives this information from the graphical user interface during simulation. *Cluster agents* also have *behaviours* for performing the analysis required for the identification of alternate source nodes etc.

VII. SIMULATION RESULTS

The *Circuit of the Future* was used to simulate all possible single faults for both the topology-independent decentralized MAS architecture and the scalable MAS architecture. The

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

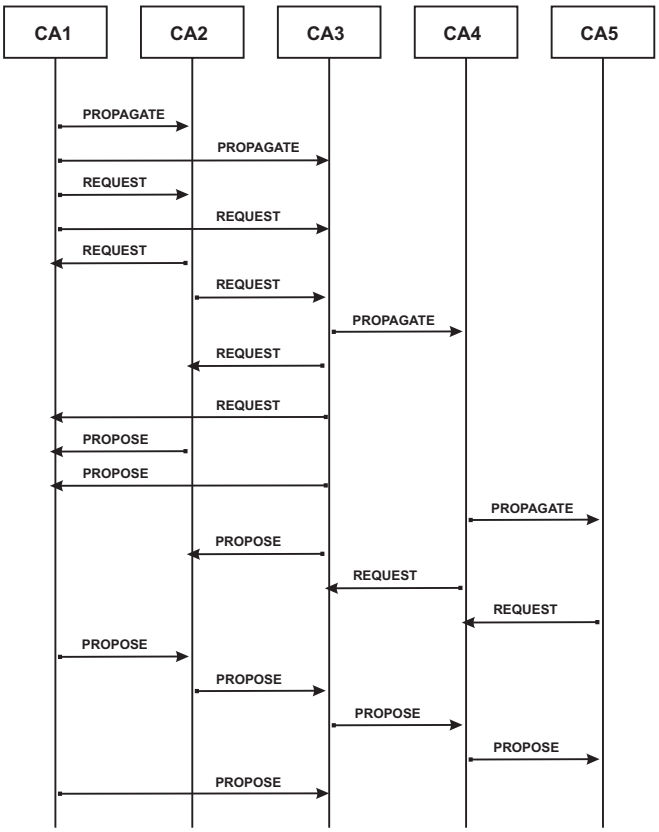

Fig. 5. Agent communication in JADE during fault scenario

results obtained from these simulations are shown in Fig. 6, Fig. 7, and Fig. 8.

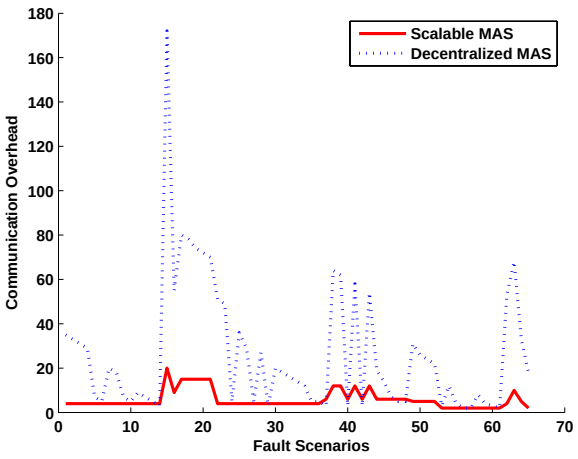

Fig. 6. Comparison of communication overhead

Fig. 6 presents a comparison of the two MAS approaches in terms of the communication overhead. We define communication overhead as the total messages passed in the agent community to reach a decision. It can be clearly seen in the figure that the scalable MAS experiences a far less communication overhead to reach the same decision as compared to the completely decentralized MAS. The average communication overhead for the scalable MAS is approximately 6 messages. On the other hand, for the decentralized MAS the average communication overhead is 27 messages. Also, the scalable MAS performance is mostly uniform, with a few occasional spikes. A more uniform performance can help to model the communication infrastructure for the system in a way that optimizes performance. On the other hand, communication overhead for the decentralized approach shows a more irregular pattern. Considering the worst case scenario, for a fault between Bus 1 and Bus 2 in the test system, the communication overhead incurred in the topology-independent decentralized MAS is 172 messages, while for the scalable MAS it is significantly lower (20 messages).

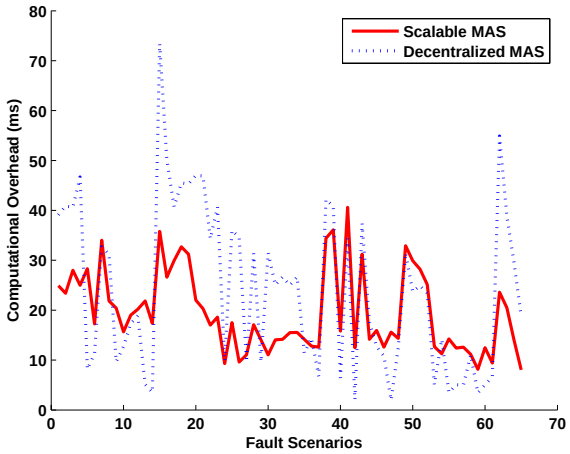

Fig. 7. Comparison of computational overhead

Next, we compare the two approaches in terms of the computational overhead. For the purpose of these simulations we define the computational overhead as the total time taken by the MAS to arrive at a decision to perform reconfiguration in the event of a fault. Multiple simulations of all possible fault scenarios were performed to obtain a comprehensive profile of the total time taken by each MAS. The results are shown in Fig. 7. It can be seen that the scalable MAS takes less total time to reach a decision. The average time taken by the scalable MAS to reach a decision is 19.47 ms, while for the decentralized MAS it is 23.92 ms. It is important to note, that not only does the scalable MAS reach a decision quickly but also, the decision is more informed. Each *cluster agent* in the scalable MAS performs a complete analysis of the situation, identifying good and bad alternate sources and based on the input from neighbors selects an optimal switching position based on the cost function and external constraints. The decentralized MAS on the other hand takes the time to propagate the fault information and any viable switching position is selected. Although the *cluster agents* in the scalable MAS are performing more work, yet the overall system is designed such that the total time taken is less. The worst case scenario in terms of the computational overhead for the

topology-independent decentralized MAS is 73.40 ms, while for the scalable MAS it is 40.60 ms.

Finally, we investigate the relationship between the total number of buses affected by a fault and the performance for each of the proposed approaches. The observations are summarized in Fig. 8. When the number of buses affected by a fault is small the scalable approach does incur some overhead. However, when the number of buses affected by a fault increases, the proposed scalable MAS outperforms the topology-independent decentralized MAS.

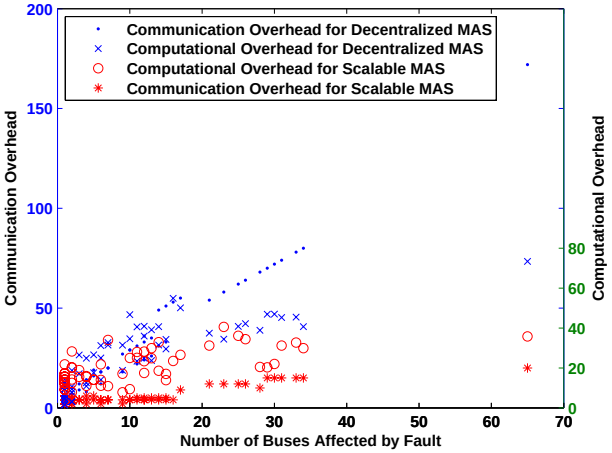

Fig. 8. Comparison of performance with respect to buses affected by a fault

We have also made some general observations about the proposed scalable MAS. Since buses affected by a fault such as loss of line in a power system are those which are physically close, they are most likely to fall in the same cluster. This reflects that the proposed choice of electrical distance as the clustering parameter is very pragmatic. One problem related to this choice can be that the electrical distances do not incorporate the system dynamics. In the future, we plan to use clustering parameters which also include information about system dynamics.

By partitioning the entire system into clusters and assigning an agent to each cluster, task sharing is achieved which reduces the problem complexity. In comparison to the topology-independent decentralized MAS, the proposed scalable MAS uses fewer number of agents and results in reducing the resource consumption and communication overhead. Also, the entire process of communication is more deterministic and less error prone. As the number of agents involved in the communication increases, the entire decision making process becomes more susceptible to faults and errors due to lost messages or problems with the communication network. Based on the simulation results and our observations and analysis, the proposed scalable MAS architecture is robust and performs well for large scale power systems.

VIII. Conclusion

Although multiagent systems and other distributed approaches offer better scalability as compared to traditional centralized approaches, yet efficient solutions based on these technologies need to scale well. Most existing decentralized multiagent system solutions for power system applications associate an agent with each component in the system. Although such solutions offer complete decentralization of control and are generally topology independent, yet for large scale systems the large number of agents in the MAS poses scalability challenges. Also, the amount of computations, communication, and coordination required for these solutions triggers the need to find a trade-off between decentralization and size of the multiagent system. Reduction in the total number of agents in a multiagent system decreases the resource requirements, simplifies the agent system topology, reduces the communication overhead by limiting the number of possible interactions, and minimizes the overall complexity of the system. The proposed scalable MAS addresses these issues by limiting the total number of agents in the system, and reducing the computational and the communication overhead. Furthermore, in the event of a fault, the affected nodes are quickly identified for reconfiguration. Depending on the system constraints, an optimal solution for reconfiguration is identified using the cost function. However, to gain these benefits, the agents in the clusters acquire additional knowledge about the topology of their cluster. Simulation results show that the proposed scalable MAS experiences significantly less computational and communication overhead as compared to the topology independent decentralized MAS. Also, the performance of the scalable MAS improves as the number of nodes affected by a fault increases, making it more attractive for large scale power systems.

Acknowledgment

This research has been sponsored in part by a grant from the US Department of Energy (DE-FC26-06NT42793).

References

[1] G. Weiss, *Multiagent systems a modern approach to distributed artificial intelligence*, The MIT Press, 1999.
[2] S. Russell, and P. Norvig, *Artificial intelligence: a modern approach* , Prentice Hall, 1995.
[3] F. Avellaneda, C. Bustacara, J.P. Garzon, and E. Gonzalez, *Implementation of a molecular simulator based on a multiAgent system* , in Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology, pp. 117-120, 2006.
[4] J. Yen, A. Chung, H. Ho, B. Tam, R. Lau, M. Chua, and K. Hwang, *Collaborative and scalable financial analysis with multi-agent technology* , in Proceedings of the 32nd Annual Hawaii International Conference, vol. Track5, 1999.
[5] K.P. Chow, and Y.K. Kwok, *On load balancing for distributed multiagent computing*, in IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 8, pp. 787-801, 2002.
[6] L.C. Lee, H.S. Nwana, D.T. Ndumu, and P.D. Wilde, *The stability, scalability and performance of multi-agent systems* , in BT Technology Journal, vol. 16, no. 3, pp. 94-103, 1998.
[7] O.F. Rana, and K. Stout, *What is scalability in multi-agent systems* , in Proceedings of the fourth international conference on Autonomous agents, pp. 56-63, 2000.
[8] S.H. Nwana, and L.C. Lee, *Stability, fairness and scalability of multi-agent systems*, in International Journal of Knowledge-Based Intelligent Engineering Systems, vol. 3, pp. 3-2, 1999.
[9] P.J. Turner, and N.R. Jennings, *Improving the scalability of multi-agent systems*, in Proceedings of 1st International Workshop on Infrastructure for Scalable Multi-Agent Systems, pp. 246-262, 2000.

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

[10] P. Scerri, Y. Xu, E. Liao, J. Lai, and K. Sycara, *Scaling teamwork to very large teams*, in Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 888-895, 2004.
[11] K. Huang, *Shipboard power system reconfiguration using multi agent system*, Ph.D. dissertation, The Florida state university, 2007.
[12] G.A. Taylor, M.R. Irving, P.R. Hobson, C. Huang, P. Kyberd, and R.J. Taylor, *Distributed monitoring and control of future power systems via grid computing*, in IEEE Power Engineering Society General Meeting, 2006.
[13] D.A. Cartes, and S.K. Srivastava, *Agent applications and their future in the power industry*, in IEEE Power Engineering Society General Meeting, pp. 1-6, 2007.
[14] H. Salazar, R. Gallego, and R. Romero, *Artificial neural networks and clustering techniques applied in the reconfiguration of distribution systems*, in IEEE Transactions on Power Delivery, vol. 21, no. 3, pp. 1735-1742, 2006.
[15] T. Brunner, W. Nejdl, H. Schwarzjirg, and M. Sturm, *On-line expert system for power system diagnosis and restoration* , in Intelligent Systems Engineering, vol. 2, no. 1, pp. 15-24, 1993.
[16] C.C. Liu, J. Jung, G.T. Heydt, V. Vittal, and A.G. Phadke, *Strategic power infrastructure defense (SPID) system a conceptual design* , in IEEE Control Syst. Mag., vol. 20, no. 4, pp. 40-52, 2000.
[17] L. Liu, K.P. Logan, D.A. Cartes, and S.K. Srivastava, *Fault detection, diagnostics, and prognostics: software agent solutions* , in IEEE Transactions on Vehicular Technology, vol. 56, no. 4, pp. 1613-1622, 2007.
[18] J.G. Gomez-Gualdron, M. Velez-Reyes, and L.J. Collazo, *Self-reconfigurable electric power distribution system using multi-agent systems*, in IEEE Electric Ship Technologies Symposium, pp. 180-187, 2007.
[19] I.S. Baxevanos, and D.P. Labridis, *Implementing multiagent systems technology for power distribution network control and protection management*, in IEEE Transactions on Power Delivery, vol. 22, no. 1, pp. 433-43, 2007.
[20] T. Nagata, Y. Tao, H. Sasaki, and H. Fujita, *Decentralized approach to power system restoration by means of multi-agent approach* , in Bulk Power System Dynamics and Control - VI, 2004.
[21] K. Huang, S.K. Srivastava, D.A. Cartes, and M. Sloderbeck, *Intelligent agents applied to reconfiguration of mesh structured power systems* , in International Symposium on Antennas and Propagation, pp. 298-304, 2007.
[22] K. Huang, D.A. Cartes, and S.K. Srivastava, *A multiagent-based algorithm for ring-structured shipboard power system reconfiguration* , in The International Conference on System, Man and Cybernetics, vol. 1, pp. 530-535, 2005.
[23] K. Huang, D.A. Cartes, and S.K. Srivastava, *A multiagent-based algorithm for ring-structured shipboard power system reconfiguration* , in IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 37, no. 5, pp. 1016-1021, 2007.
[24] K. Huang, S. Sanjeev, and D. Cartes, *Decentralized reconfiguration for power systems using multi agent system* , in Proceedings of the 1st Annual 2007 IEEE Systems Conference, pp. 253-258, 2007.
[25] K. Huang, S.K. Srivastava, D.A. Cartes, and M. Sloderbeck, *Intelligent agents applied to reconfiguration of mesh structured power systems* , in International Conference on Intelligent Systems Applications to Power Systems, pp. 1-7, 2007.
[26] F. Ponci, and A.A. Deshmukh, *A mobile agent for measurements in distributed power electronic systems* , in IEEE Instrumentation and Measurement Technology Conference, pp. 870-875, 2008.
[27] D. Elizalde, D. Staszesky, and M. Meisinger, *Use of distributed intelligence for reliability improvement using minimum available distribution assets*, in IEEE/PES Transmission and Distribution Conference and Exposition, pp. 1-6, 2006.
[28] D.M. Staszesky, *Use of virtual agents to effect intelligent distribution automation*, in IEEE Power Engineering Society General Meeting, 2006.
[29] Z. Li, X. Chen, K. Yu, B. Zhao, and H. Liu, *A novel approach for dynamic reconfiguration of the distribution network via multi-agent system*, in 3rd International Conference on Deregulation and Restructuring and Power Technologies, pp. 1305-1311, 2008.
[30] S.L. Hamilton, C.K. Vartanian, M.E. Johnson, A. Feliachi, K. Schoder, and P. Hines, *Circuit of the future: interoperability and SCE's DER program*, Bulk Power System Dynamics and ControlVII. Revitalizing Operational Reliability 2007 iREP Symposium, pp. 1-9, 2007.
[31] N. Muller, and V.H. Quintana, *A sparse eigenvalue-based approach for partitioning power networks* , in IEEE Transactions on Power Systems, vol. 7, no. 2, pp. 520-527, 1992.
[32] S.E. Schaeffer, *Graph clustering*, in Computer Science Review, vol. 1, no. 1, pp. 27-64, 2007.
[33] U. Luxburg, *A tutorial on spectral clustering* , in Statistics and Computing, vol. 17, no. 4, pp. 395-416, 2007.
[34] L. Zelnik-Manor, and P. Perona, *Self-tuning spectral clustering* , in Adv. Neural Inf. Process. Sys., 2004.
[35] A. Ng, M. Jordan, and Y. Weiss, *On spectral clustering: analysis and an algorithm*, in Advances in Neural Information Processing Systems 14, 2001.
[36] P.M. Anderson, and A.A. Fouad, *Power system control and stability* , IEEE Press, 2002.
[37] F.L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*, Wiley, 2007.

Open Science Index, Computer and Information Engineering Vol:3, No:3, 2009 publications.waset.org/4665.pdf

Summiya Moheuddin is currently pursuing her M.S. in Computer Science at the Lane Department of Computer Science and Electrical Engineering at West Virginia University. She is a Graduate Research Assistant and has been a member of the Advanced Power & Electricity Research Center at WVU since 2007. She received her B.S. degree in computer science from National University of Computer and Emerging Sciences-FAST, Pakistan in 2006. She worked as a software engineer from 2006-2007. Her research interests include distributed artificial intelligence, distributed computing, knowledge discovery techniques, and reliable computing.

Afzel Noore received his Ph.D. in Electrical Engineering from West Virginia University. He worked as a digital design engineer at Philips India. From 1996 to 2003, Dr. Noore served as the Associate Dean for Academic Affairs and Special Assistant to the Dean in the College of Engineering and Mineral Resources at West Virginia University. He is currently a Professor and Associate Department Chair in the Lane Department of Computer Science and Electrical Engineering. His research interests include computational intelligence, biometrics, software reliability modeling, machine learning, hardware description languages, and quantum computing. His research has been funded by NASA, NSF, Westinghouse, GE, Electric Power Research Institute, the US Department of Energy, and the US Department of Justice. Dr. Noore has over 95 publications in refereed journals, book chapters, and conferences. He has received four best paper awards. Dr. Noore is a member of the IEEE and serves in the editorial boards of Recent Patents on Engineering and the Open Nanoscience Journal. He is a member of Phi Kappa Phi, Sigma Xi, Eta Kappa Nu, and Tau Beta Pi honor societies.

Muhammad Choudhry received his B.Sc. degree in Electrical Engineering from the University of Engineering and Technology. Lahore. Pakistan in 1973. He received his M.S. (EE) from University of Kansas in 1977 and Ph.D. degree from Purdue University in 1981. From August 1973 to December 1975, he was Assistant Engineer with Water and Power Development Authority in Pakistan. He joined West Virginia University in 1981 and is currently Professor in the Lane Department of Computer Science and Electrical Engineering. His areas of interest are Multiterminal HVDC System, System Stability, Optimal Control and System Identification.