Multi Agent System for Energy Management

W.D.N. Fernando, K. D. Dhanuka Lakmal Kodikara, Piyasena T.T.S, Randana G.A.R and K.A.D.N. Kishan Wimelawarne
Department of Computer Science & Engineering, University of Moratuwa, Sri Lanka.

***Abstract*-Multi Agent Systems is an evolving field of Computer Science which is yet to be explored and widely being explored due to its fascinating capabilities.**

On the other hands, Electrical energy is one of the most widely used sources of energy in the world today. Its uses are highly diverse ranging from the high speed railways of France to the common everyday heaters. Thus, it becomes more than a need to utilize it in an efficient and economic manner.

Despite its heavy usage, we can find a common pattern in the domestic usage of electricity where extreme high usages occur for short durations.

Our attempt is a combination of the above two topics where we try to use a Multi Agent Network to manage the electrical appliances in such a way that the high peaks are flattened out through the time which brings many economic benefits.

This document clearly indicates our approach and how we integrated the above systems to attain the desired goal.

We are highly grateful for the effort and commitment by Maheendra Piraveenan* in helping us throughout the project in giving us advice and guidance regarding all issues.

I. Introduction

Our Aim is to implement a MAS (Multi Agent System) which can cooperate with the real world electrical devices in such a way that the high peaks which occur for short durations are flattened out with respect to time.

The initial research concern which arises is the practicability of reducing peaks by distributing the load with respect to time. It may seem impracticable at the first glance. When we request power for a device, we expect to turn it on. We cannot wait for some time till it turns on. But one aspect which we do not consider is that there exist many power consuming devices which can operate for lengthy time durations without drawing power from the power lines. One such example is a refrigerator. Once cooled down to the minimum temperature, it can stay for long durations without acquiring power from the grid.

Our concept is based on such devices which have the capability of accumulating energy. By having an Intelligent Agent,(which can predict the energy needs for the future) embedded into such devices, we can collaborate with the other Agents. In doing so, we can have a schedule of energy consuming time slots which each device agrees upon. This way we can reduce the peaks by reducing the possibility that each device tries to acquire power at the same time. Hence, we can flatten the Electrical Energy usage curve.

One of the major issues regarding this concept is the final result which we obtain in implementing the system. It may not seem obvious. But due to the high power usage, we need to spend a great amount of initial infrastructure costs to establish the system. Due to the high usage of Electricity within a few hours, we need to build the system capable of handling the maximum load. This is a huge increase in expense.

As we mentioned, Multi Agent Systems are very effective means of implementing such a network due to many of its features. Multi Agent Systems are by nature scalable, robust and dynamically expandable. We attach each device with an Agent who can calculate the device's future energy demand by considering its present state. By having a network of such Intelligent Agents, we define a protocol which each Agent must adhere to. When adhering to the protocol, the total system would behave and use the energy in a manner to avoid high peaks.

As discussed, conceptually it is possible to build a Multi Agent System which would coordinate with each other to bring down the peak power usage by distributing the load with respect to time. Our research is based on proving the above concept by implementing such a Multi Agent System and hence exhibiting its feasibility.

*(Affiliated to : Department of Computer Science and Engineering, University of Moratuwa and School of Information Technologies, University of Sydney, Australia)

II. Overview Of Agent Network

The most critical component of a Multi Agent System is its network architecture. Our purpose of acquiring a Multi Agent System for energy management is to overcome the current situation of centralized controlling of power supply. The proposed architecture therefore should withstand failures of agents. As well there should be no single point of failure for the system. Furthermore it should be as flat as possible meaning that there should be none, or minimum level of hierarchies. By considering all these factors, and examining existing Multi Agent Systems, [1]-[4] we devised an agent network architecture that is very reliable and robust as described below.

Our agent network consists of four types of agents denoted by

SINK AGENTS – Agents running at power sinks

SOURCE AGENTS – Agents running at power sources

SINK BROKER AGENTS – Agents who coordinate the *Sink Agents* to optimize their power consumption. These agents normally run at power distribution sub stations. Each *Sink Broker Agent* has a substation id, and each substation may contain several *Sink Broker Agents* for load balancing activities. The agents which run at the same substation contain the same substation id

SOURCE BROKER AGENTS – Agents who coordinate the *Source Agents* to supply their power in most economical way (Among the *source broker agents* one agent serve as a leader for coordination. The leader selection algorithm and how it withstands the failures are described in the Section VI A)

Overview of our proposed agent network is illustrated in fig. 1. The total network works as a interconnection of two networks named as *SINK NETWORK* and *SOURCE NETWORK*. *Sink network* manages the power optimization of *Sink Agents* and *source network* manages the power supply of the *source agents* economically. In order to maintain the smooth operation of these networks and their interconnection, several algorithms were developed. Those Algorithms are described in Section IV B of this paper.

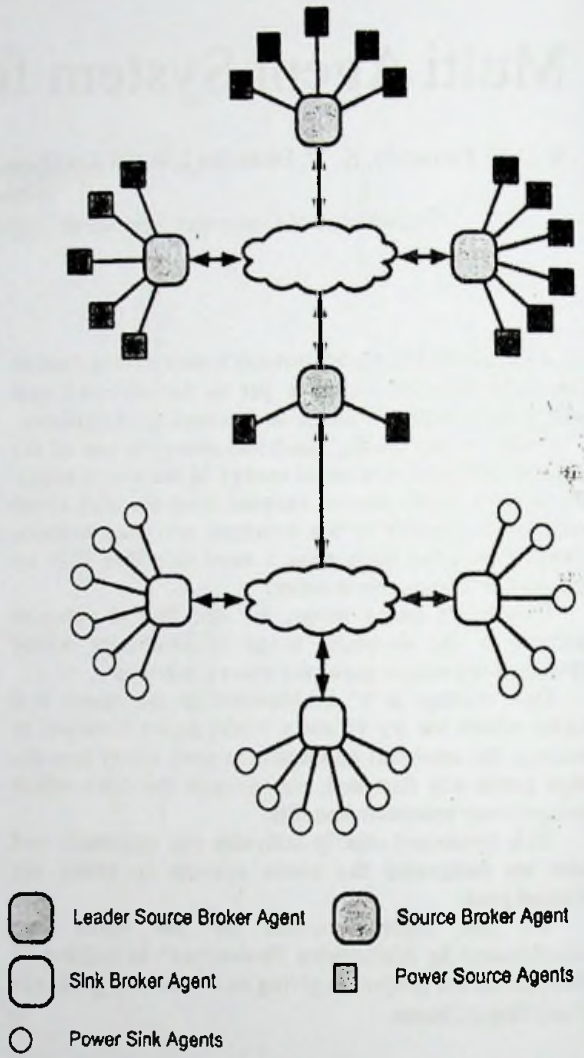

Fig. 1. Agent Network overview.

III. Sink Agent – Sink Broker Agent Connection Establishment

When a *Sink Agent* is added to the Agent network, it should initially establish a connection with an existing *Sink Broker Agent* within its substation area. This procedure must occur prior to any other activities. Hence, as the first task, the new *Sink Agent* finds all the *Sink Broker Agents* running in the network within the *Sink Agent*'s substation. It then sends separate requests to each of them indicating its power rating. Each *Sink Broker Agent* calculates its present amount of power distribution and sends back the response to the *Sink Agent* who made the request. The *Sink Agent* selects the *Sink Broker Agent* who sends the minimum value and connects with it to obtain power[1]. This value is calculated in a manner such that, the lower the load a

Sink Broker has, the lower the response value is. Hence, we try to distribute the load with respect to Sink Brokers Agents. This procedure is taken to ensure a near equal power distribution among the *Sink Broker Agents* without allowing a single *Sink Broker Agent* to obtain a considerable amount of power from the system.

A. Maintain a reliable Sink Agent – Sink Broker Agent connection

Maintaining the reliability of the *Sink Agent – Sink Broker Agent connection* is a vital role in our proposed agent system. In a distributed environment some agents may crash and fail unexpectedly. If a *Sink Broker Agent* fails, all its *Sink Agents* have no means of obtaining power. To address this issue, all the *Sink Agents* regularly check the status of their *Sink Broker Agent* via messages. If the *Sink Broker Agent* does not respond, the *Sink Agents* can simply find another *Sink Broker Agent*. Furthermore, mechanisms have been implemented for all the Agents to send termination messages to their dependent Agents in an event of failure. Hence, the dependent Agents can safely take measures to avoid their failure. This method may not work in some scenarios when sudden crashes occur. Thus, regular checking has been included for the Agents in order to withstand such critical situations.

IV. Sink Power Optimization

The proposed power optimization algorithm runs within the *Sink Broker Agents* and *Sink Agents*. The details of the algorithm had been discussed in Section IV B. In this algorithm, we use *Power Plans* of the Agents to develop the algorithm. Before the discussion of the algorithm, it is required to define exactly what is meant by the terminology *Power Plan*, and some operations defined on them. Section IV A clarifies the definitions.

A. Definition of a Power Plan

In the system, we define a *Power Plan* as a collection of a finite number of *Plan Elements*. A *Plan Element* is a positive value representing the power consumption of a device at that point in time. Each *Power Element* has a fixed and same value for the *Element Length* across the whole *Power Plan*. Hence we represent a *Power Plan P* of n plan elements as follows.

$$P = ((p_1, p_2, p_3 \dots p_n), d) \quad .$$

Where $p_1, p_2, p_3 \dots p_n$ represent plan elements and d is the element length

Total energy of the plan P can be defined as follows

$$Energy(P) = d \times \sum_{i=1}^{n} p_i$$

Peak Power of a plan P is denoted by

$$Peak(P) = \max(p_i \mid i = 1 \dots n)$$

Average Power Consumption of plan P can be denoted by

$$Average(P) = \frac{E(P)}{nd} = \frac{1}{n} \times \sum_{i=1}^{n} p_i$$

Let P and Q be two separate power plans. Addition of power Plans P and Q is defined as follows. In order to add two *Power Plans*, the element count and the element length of both plans should be equal.

$$P = ((p_1, p_2, p_3 \dots p_n), d)$$

$$Q = ((q_1, q_2, q_3 \dots q), d)$$

$$P + Q = ((p_1 + q_1, p_2 + q_2, p_3 + q_3 \dots p_n + q_n), d)$$

B. Sink power optimization algorithm

Several existing solutions for distributed planning [5]-[7] were analyzed to arrive at our proposed algorithm. At the initial phase of the algorithm there is a set of power plans $\{P_1, P_2 \dots P_n\}$ received by the *Sink Broker Agent* from its *Sink Agents*. The proposed algorithm models this optimization problem as a constraint satisfaction problem. The specialization of the proposed algorithms for distributed constraint satisfaction problem by Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara [5] is considered in building the proposed solution for power optimization. The constraint for the problem is constructed as follows[2].

a. Note that a *SINK AGENT* connects with a *SINK BROKER AGENT* does not imply that it establish a physical connection with that broker. It simply means that the *SINK AGENT* has decided to communicate with the specified *SINK BROKER AGENT* to fulfil its power requirements.

b. Limit is a value greater than 1. The final optimization will be very good when the limit is closer to 1, but may take considerable time to be optimized. When the value of limit is considerably greater than 1, then optimization will not be that good but will be finished quickly. The value of the limit should be selected according to the status of the type of the agent network.

$$Peak\left(\sum_{i=1}^{n} P_i\right) \leq Average\left(\sum_{i=1}^{n} P_i\right) \times limit$$

Constraint 1

During the optimization process, several types of messages are interchanged amongst the agents. The types of the messages and its purposes are defined as follows.

initMessage – sent by the *Sink Broker Agent* to all its *Sink Agents* to inform the beginning of the next optimization cycle

initialPlanRequestMessage – sent by the *Sink Broker Agent* to all its *Sink Agents* for requesting the first plan

planReviceMessage – sent by the *Sink Broker Agent* to some selected *Sink Agents* for requesting of revised plans

sendPlanMessage – sent by the *Sink Agent* to its *Sink Broker Agent* for sending its power plan

At regular time intervals of T minutes, the *Sink Broker Agent* sends all its *Sink Agents* the *initMessage*. At the arrival of *initMessage*, each *Sink Agent* applies the previously agreed *Power Plan* to control their devices and start generating new *Power Plans* for the next plan cycle[3].

The details of how Agents generate their plans are described in Section V. After a while, the *Sink Broker Agent* sends all its *Sink Agents* the *initialPlanRequestMessage*. Then, upon the arrival of *initialPlanRequestMessage, Sink Agents* sends one of its generated plans to the *Sink Broker Agent* via the *sendPlanMessage*. The *Sink Broker Agent* collects all the initial plans sent by the *Sink Agents* and evaluates the constraint specified by (constraint 1). If the constraint is satisfied then the plans are already optimized and algorithm terminates as the constraint is met.

If the constraint is violated, then the *Sink Broker Agent* finds the suitable *Sink Agents* (criteria to find the suitable *Sink Agents* is described in the Section IV B 1) to send the *planReviceMessage* and send *planReviceMessage*.. At the arrival of *planReviceMessage, Sink Agents* revise their plan and sends a different plan (which was calculated before) to the *Sink Broker Agent* via the *sendPlanMessage*. Once more, the *Sink Broker Agent* collects all the responses and re-evaluates the constraint specified by (constraint 1). The same procedure is followed until the plans satisfy the specified constraint.

By following these procedures Agents will ultimately come to situation where the constraint is satisfied. Then the algorithm will terminate. And the whole process would start again after T minutes.

A specific value for T cannot be specified. The value of T should be selected based on the amount of devices and scale of the network. For demonstration purposes by considering the size of the system, T was selected with a value of 15 minutes. (With $T=15$, we were capable of getting successful results) For ultimate optimization of a network, the optimum value must be calculated by testing specifically for the system.

1) Selecting Sink Agents to send planReviceMessage

Sink Broker Agent calculates the total of all the arrived plans.

$$T = \sum_{i=1}^{n} P_i$$

Calculate the threshold value as

$$Average(T) \times limit$$

Then, the *Sink Broker Agent* finds the peaks of the total plan (T) which are higher than the threshold value. The example scenario for three *Sink Agents* is illustrated in fig. 2 for describing purposes. According to the figure, *Plan Elements* 1, 3, 5 and 6 are greater than the threshold value. Then for each such peak, the *Sink Broker Agent* finds the minimum contributing sink for that peak. (As in the figure, minimum contributor for the peak at element 1 is sink 3) These minimum contributors are selected to send *planReviceMessage*.

The criteria for selecting the minimum contributing *Sink Agents* can be described as follows. Suppose, we select the highest contributing *Sink Agent* at a conflicting time element.(As in the figure, it is Sink1). If we request Sink1 to send a different *Power Plan* by *planReviceMessage*, there is a very high probability Sink1 needs that quotient of power and it would request the same value at a different time element. Thus, it would create a peak at a different time. Hence, there is a high probability that it would create a peak which did not exist before. Therefore, it would be difficult to obtain our aim. But, if we send the lowest power requesting Sink with a *planReviceMessage*, then, there is a high degree of probability that it may squeeze into a time element without creating a peak.

c. If there is no previously built plan, then the agent will allow the device to control itself, Agent start to control it from the next plan cycle.

As in the example, we could safely relocate Sink3 to obtain power at time element 7 without violating the condition. Hence, we send *planReviceMessage* to the lowest power requesting Sink at the conflicting time element.

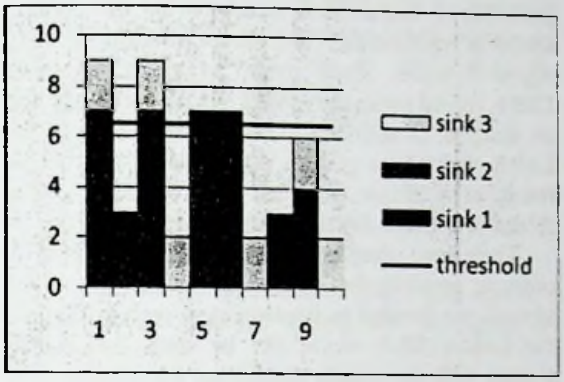

Fig. 2. Total plan with threshold for three sink agents.

V. Power Plan Generating Algorithm

When we consider a *Sink Agent*, it needs to generate *Power Plans* for the device it is controlling for the next dn time period. That is, If we consider *Power Plan* $P = ((p_1, p_2, p_3 \dots p_n), d)$, the respective *Agent* should be capable of generating *Power Plans* for the device for the time duration $t_0 + dn$ to $t_0 + 2dn$, at t_0. And also if we consider *Plan Element* p_i and let the commencing time of the *Plan Element* be t_{pi}. p_i is calculated as $p_i = max(p_c)$, where p_c is the power consumed by the device which is controlled by the agent for the time duration t_{pi} to $t_{pi} + d$. With these considerations we can describe plan generating algorithm as follows.

As an example, consider the device required to control as a cooling room with following configuration.

T_{ao} - External Air Temperature
Q_p - Power Rating
Q_{pi} - Idle State Power
C_a - Thermal Capacity
U_w - Wall Thermal Resistance
U_v - Ventilation Path Thermal Resistance.

Then, the internal temperature of the cooling room is governed by the equation [3],

$$T_{ci} = \frac{1}{C_a f + k}\left(C_a f T_{ci}(t-1) + Q(t) + k T_{ao}(t)\right)$$

Where, T_{ci} is the internal temperature of the cooling room, f is the sampling rate, $k = A_w U_w + U_v$ and Q is power consumed by the cooling, A_w is a coefficient.

Since the cooling room has only two power consuming states, we can consider plan element p_i of the power plan P for the cooling room as follows,

$$p_i = \begin{cases} Q_p, & \text{when cooling room is cooling} \\ Q_{pi}, & \text{when cooling room is not cooling (idle)} \end{cases}$$

Then, if we consider power plan $P = ((p_1, p_2, p_3 \dots p_n), d)$ with n elements there will be 2^n number of power plans for cooling room device for one dn time period. Let S_A denote this power plan set, then

$$S_A = \{P_i \mid i = 0 \text{ to } 2^n\}$$

Let the upper temperature limit of the cooling room be denoted by T_u and lower temperature limit by T_l. Then, the refrigerator agent will compute no of power plan elements, i where each element $p_i = Q_p$ to bring cooling room temperature T_u to T_l and no of power plan elements, h where each element $p_i = Q_{pi}$ to bring cooling room temperature T_l to T_u, when refrigerator agent is switched on.

Then cooling room agent will sense cooling room parameters and identify plan set S_i where

$$S_i = \{P \mid P \in S_A, P = (p_1, p_2 \dots, p_i, \dots, p_n)$$
$$\text{where } p_j = Q_p \; \forall_{j=i,\dots,i+k} \text{ and } 1 \le k \le n - (i+1)\}$$

$$S_h = \{P \mid P \in S_A, P = (p_1, p_2 \dots, p_i, \dots, p_n)$$
$$\text{where } p_j = Q_{pi} \; \forall_{j=i,\dots,i+k} \text{ and } 1 \le k \le n - (i+h)\}$$

Power Plans in S_i are not suitable for the device since they violate the conditions of the cool room. And also *Power Plans* in S_h are not suitable for the device since they violate the basic operation temperatures for the device.

Therefore all the possible set of power plans for the cooling room is given by

$$S_p = S_A - S_i - S_h$$

Then, when the refrigerator receives its first *initmessage* from the *Sink Broker Agent*, it will sense its internal temperature of the cooling room and calculate internal temperature at the end of first nd time duration using the above model.

Then, take each plan $P \in S_p$ and check whether P is usable for next nd time duration and generate plan set S_u. For each $P \in S_p$, the Agent will check whether following constrain, C_1 is satisfied

$$T_i < T_{ai}(t_0 + id) < T_u$$

where t_0 is plan start time and $i = 1 \ldots n$

Then

$$S_u = \{P \mid P \in S_p, P \text{ satisfies } C_1\}$$

The refrigerator agent will sort the plans of the S_u according to the $Energy(P)$ where $P \in S_u$ in descending order.

The Agent will send its first plan of S_u for the *Sink Broker Agent* when plan request message is received from the *Sink Broker Agent*.
The *Sink Agent* will send its next *Power Plan* if any plan revise message is received from the *Sink Broker Agent*.

VI. Source Broker Agent (SBA) Algorithms

Algorithms which are related to SBAs (Source Broker Agents) as well as *Source Agents* registered with the system are described under this Section.

By doing several researches, we were capable enough to find high performance and effective algorithms suitable for controlling the above mentioned Agents defined in *MASEMA*. Under this, we mainly focus on the following two algorithms.

1) Leader Source Broker (LSB) Finding Algorithm
2) Power Sources Management Algorithm

Advantages and disadvantages of the respective algorithms will be discussed under their subtopics.

A. Leader Source broker finding algorithm

Unlike the Sink Broker Agents, the Source Broker Agents (SBAs) have a leader Agent. Despite the name and task associated with the leader, any of the SBAs can become the leader. In other words, the leader is one of the Source Brokers (SBs) who has been designated as the leader. The leader changes dynamically and thus is non deterministic on who will become the next leader.

It would have been a simpler approach to simply assign one Leader Source Broker Agent (LSBA) for all of the SBs. But this would be a bad design in terms of Multi Agent Systems. Furthermore, it would introduce a hierarchical structure on the System which is regarded as a bad design in terms of Multi Agent Systems. Even more, having a dedicated LSBA would mean the system would highly depend on that for its activities. Thus, in the failure of the LSBA, the whole system would crash. This is the result of a single point of failure which is not tolerable in Multi Agent Systems.

Therefore, despite the extra difficulties, and in order to preserve the integrity and flexibility of the system, we decided to implement in such a way that the Leader SBA would not be static and would change with the system conditions. Furthermore, the advantages we got over the extra effort were very high.

We get several disadvantages in implementing the system with a LSBA in comparison to a system without a LSBA. However, in doing so, we get a lot of advantages regarding many factors of the system. The advantages we get over doing it so outweighs the disadvantages associated with it. In such scenarios, we mitigate the effects of having a LSBA by making it dynamic by nature. Thus, we preferred to implement the system having a LSBA. Furthermore, the impact made by the disadvantages associated is of minimum.

1) Advantages

i. If we did not have a LSBA, then all the *Sink Broker Agents* would need to send individual requests to all of the SBAs. This would create confusion amongst the *Sink Broker Agents*. In doing so, we utilize unnecessary bandwidth of the system.

ii. In having a LSBA, we can have better results than all individual SBAs doing the calculations.

iii. If not, all the *Source Broker Agents* would need to perform the calculations regarding power sources. This would be a redundant amount of calculations. Thus, now it is done by only one.

iv. The leader is selected on a criteria based on the number of Source Agents connected. Thus, the lesser the number of Agents connected, the higher the probability of it becoming the leader. Hence, the leader selected has minimum work regarding Source Agents. In doing so, we avoid the risk of the Leader being overloaded with work.

v. A function continuously monitors on which *Source Broker Agent* should become the next leader. Thus, if a better option is available, it changes the leader.

vi. Even though the leader terminates unexpectedly, the others would carry on their work by assigning a new leader of the remaining SBAs. Hence, by introducing a leader, we do not violate the robustness of the network.

2) Disadvantages

- The work load on the LSB may become relatively high at times.
- The LSB does all the calculations and acts as a manager or dictator asking and managing the subordinates.

3) Algorithm

Lead Source Broker Agent is identical to any other SBA in terms of code and functionality. There is no special class for LSBA. All the *Source Broker Agents* are derived from the same class. The Leader is just an attribute with respect to time. You cannot judge who would be the next leader by prediction.

If there is only one SBA, then it would be assigned as the LSBA by default. All other *Source Broker Agents* would need to register themselves with the LSBA. The LSBA would have connections with each and every SBA.

Through our research, we observed that 30 minutes would be a good value for the LSB to check for the next LSB. We could have used it to be a lower value than 30 minutes. But, if the value is very small then we would face the following problems,

i. When the time is small, there would be a lot of transitions of the LSBA. Therefore, it would reduce the performance and introduce conflicts in passing messages.

ii. When we introduce a low time interval, it becomes evident that the system would be doing more work on translating information gathered by one Leader to the other and backwards.

iii. The disadvantage of not changing the leader rapidly is minimized.

iv. The sole purpose of changing the Leader is to provide a distributed and balanced load amongst each SB. Better performance is not gained by reducing the time of changing the LSBA.

Every 30 minutes, (The value is not hard coded, it can be easily changed by simply editing the properties file) the current LSBA would send messages to all of the other SBAs requesting them to send a value which is used to decide the next suitable Leader Broker.

Each SBA then calculates a value V based on the following algorithm.

i. Let there be n number of SBAs

ii. Let B_i be the i^{th} SBA

iii. Let x_i be the number of Agents connected to the i^{th} SBA

iv. V_i is calculated as follows

v. Let C denote the cost associated with one Source Agent

$$V_i = x_i \cdot C + 1000$$

Upon calculation of the above value, each SBA would send the response to the present LSBA. The LSBA would be assigned as the SBA with the minimum V_i value. We can further state the following,

--If $V_i < V$ *of present Leader* $\Rightarrow B_i$ would be the next LSBA.
--Else, the present leader would remain as the leader for the next iteration.

The reason behind assigning the SB with minimum devices connected is to reduce the work load on the LSBA. The SBA with the minimum number of Agents connected is obviously doing the least amount of work. Thus, it is the ideal candidate to be selected it as the Leader.

Once a new LSBA is assigned, the other SBAs would connect to the new LSBA. It simply means that once some becomes new leader he must keep the existing SBAs list and must send message to other SBAs that he is the new leader.

Once, a SBA disconnects itself from the LSBA, and then the leader would remove the entry from its list of other SBAs.

Once the existing LSB terminates, the other SBAs would find the next LSBA from the above mentioned algorithm.

B. Power Sources Management Algorithm

Each power Source Agent would contain the following parameters,

--Agent ID
--Maximum power which can be supplied
--Cost of supplying the power
--Whether the power plants are up and running or not

Each SBA maintains a table consisting of the above values. This table is updated every T minutes. (At present, we have selected $T=2$ for demonstration. A suitable value needs to be determined for the system depending on the scale of the system). If any changes occur regarding these parameters, the respective Source Agent would inform the respective SBA informing of the change. Then, the SBA would update its table accordingly.

The LSBA would continuously keep an updated table of the available power sources and the costs associated with each power source. This table is similar to the table kept by the other SBAs.

The Sink Brokers inform their power needs to the LSBA. The Sink Brokers would do so once they have finalized their power plan for the next t (we have set $t=2$ for demonstration. A suitable value needs to be calculated for the system by testing) minutes. Every T minutes, the LSBA checks the demand of electricity of each of the *Sink Broker Agents*.

The LSBA then calculates the total demand for the *Sink Broker Agents* for the next T minutes duration. The LSBA calculates the power available by the SBAs. This is done by referring to the power source table values.

If the demand is less than the availability, then the LSBA does not perform any operations to turn on additional power sources (But actually, we keep a margin for error and thus always try to keep availability at an additional factor than the demand. This will be discussed in Sub Section VI B 1).

However, if the demand for the next T minutes is greater than the availability, then the LSBA would request the SBAs to turn on additional power plants. Every t minutes, the LSB updates itself by getting data from the SBAs.

As described earlier, the LSBA keeps an updated table of all of the power sources and the maximum power available along with the cost and state which describes whether the power source is up or down. This table is arranged in an order such that the minimum cost power sources are at the top. Similarly, high cost power sources are at the bottom.

Now suppose the demand goes higher than the present supply. Then, the LSBA would inform sources to be ready to switch them on in the next iteration in the order given above.

1) Need of keeping extra availability of power

As described, we actually keep a limit above the necessary demand of power at all times. This is done with the following reasons in mind.

i. In order to preserve the dynamic nature and ad hoc capability of the system, we have introduced some margins for errors by allowing new devices to be connected at any time.

ii. In order for a device to run separately, regardless of the failure of its Agent, we have let the ability of that device to operate as if it was connected to the electricity grid without an intelligent Agent.

iii. Before an Agent establishes a connection and calculates plans, the actual device needs to operate. Thus, we allow it to operate as usual during the transition period. This preserves the availability of the device.

iv. This is done by maintaining the availability above the required limit. Suppose the present demand on electricity is 100 W. Suppose the tolerance percentage is 15 %. Then, we would make sure that the total available power to the system is at least 115 W. Therefore, even if some new devices attempts to connect to the system, they would not have any scarcity of power. (For demonstration purposes, we have used 15% factor as additional factor.)

v. Furthermore, we do predictions on the future demand by considering the present status. We cannot guarantee 100% of accuracy of the predictions. They are very much capable of changing. Thus, it is more than a need to keep an extra amount of power.

vi. Sometimes, there may be unexpected changes which may occur to a device which changes its predictions and calculations. As an example think of a heater room. Under normal conditions, it may not lose its heat for 30 minutes. So, when the Sink Broker Request to send its Energy plans, the Heater Room sends the plan as zero for the next 15 minutes. But suddenly, the operator may open the door for several minutes, loosing thermal energy. Thus, the Heater Room may become in need of heating sooner than 15 minutes. Say 7 minutes. Thus, after 7 minutes, the Heater Room would start drawing power from the national grid.

vii. Also think what would happen if the operator selects a new temperature range for the Heater Room. i.e. Suppose the old requirement was to maintain the room between

   i. Lower limit : 125 ^{0}C
   ii. Upper Limit : 200 ^{0}C

iii. Present Temperature : 150 °C

In order to reduce from 150 °C to 125 °C, it may take 30 minutes for the Heater Room. So, for the next 15 minutes, it did not request power. But suddenly the operator turns the minimum temperature to 175 °C. Then, the Heater Room should start to heat immediately. Thus, the device starts to draw power from the national grid. Hence we must maintain a margin for such situations.

Under such circumstances, the main objective is to supply power to the device. If not, then the objective of the system would not be met. It is to be noted that the aim of the research is not to save energy. But, distribute the energy consumption time periods with minimum peaks. Hence, we cannot stop it from receiving power from the grid. If we do, then we would basically starve a device which is not the aim of the research. Therefore, we must maintain a value above the required limit for safety issues such as the above mentioned.

VII. Conclusion

The Aim of the research was proving the capability of reducing the peak power consumption of an Electrical network by distributing the load with respect to time using a Multi Agent System. In order to do so, we devised several algorithms and protocols which we could use to distribute the load evenly through time. By abiding to the protocol, each Energy consuming Agent would collaborate in a manner which would in turn reduce the peak load.

We are able to conclude, that by using the MAS(Multi Agent System) architecture mentioned in Section II, we were able to obtain a reduced peak value of the load with respect to time. By using the concept of Power Plans and by using the optimization algorithms, we were able to distribute the load with respect to time and thus attain the desired Aim of the research.

Fine tuning of the MAS has not been attempted by our research. It would be the result of another series of researches followed by excessive testing procedures. Our Aim was to prove that the theoretical concept of reducing the peak power by distributing the load was possible. Through the research we were able to conclude that we achieved our Aim.

Our research was based on some of the least explored areas of computing. There are no established procedures for performing specific tasks. Hence, we had to device some of the procedures ourselves. Such procedures may lead to controversy. But, it must be noted that all such conflicting decisions have been fully justified.

In speaking of the applicability of MASEMA (Multi Agent Systems for Energy Management) to the real world applications, we see that it cannot be done within a day. There needs to be a solid foundation. A considerable amount of changes need to be done. We admit, the system utilizes a significant amount of network bandwidth. This may not be feasible today. But, in the future, as the technology rapidly improves, such technological improvements would nurture the development of this concept into practise.

Science and Technologies experience a rapid growth with time. Hence, what was not feasible in the past had become a reality nowadays. Likewise, Multi Agent Systems and communication media are bound to experience such growth in the near future. In the future when a power crisis is likely to occur, with the development of Multi Agent Systems and related technologies, MASEMA would have a vital role to play in distributing power in an efficient manner. If so, refrigerators, televisions and other electrical appliances may come with factory built Intelligent Agents which can be directly integrated to the MASEMA network.

. References

[1] N. R. Jennings, J. M. Corera, I. Laresgoiti, *"Developing Industrial Multi-Agent Systems"*

[2] Jiaming Li, Geoff Poulton, Geoff James, Astrid Zeman,Peter Wang, Matthew Chadwick, Mahendra Rajah Piraveenan, *"Performance of Multi-agent Coordination of Distributed Energy Resources"*, January 2007, WSEAS International Conference on Computer Engineering and Applications.

[3] Rongxin Li, Jiaming Li, Geoff Poulton and Geoff James, *"Agent-Based Optimisation Systems for Electrical Load Management"*

[4] Shadi Abras, St'ephane Ploix Sylvie Pesty Mireille Jacomino, *"A Multi-Agent Home Automation System For Powermanagement"*

[5] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara, *"The Distributed Constraint Satisfaction Problem: Formalization and Algorithms"*, Octomber 1998, IEEE Transactions On Knowledge And Data Engineering.

[6] Cesar Fernandez, Ramon Bejar, Bhaskar Krishnamachari Carla Gomes, Bart Selman, *"Communication And Computation In Distributed CSP Algorithms"*

[7] Edmund H. Durfee, *"Distributed Problem Solving and Planning"*