Resource coordination in wireless sensor networks by cooperative reinforcement learning

Muhidul Islam Khan
Institute of Networked and Embedded Systems
Alpen-Adria Universität Klagenfurt
Klagenfurt, Austria
e-mail: muhidulislam.khan@aau.at

Bernhard Rinner
Institute of Networked and Embedded Systems
Alpen-Adria Universität Klagenfurt
Klagenfurt, Austria
e-mail: bernhard.rinner@aau.at

Abstract— **Wireless Sensor Networks (WSN) typically operate in dynamic environments, hence we can not schedule the execution of tasks a priori. This must be done online in a way to minimize the resource consumption. We present a cooperative reinforcement learning approach to schedule the tasks in WSN. A WSN is composed of a large number of tiny sensing nodes capable of interacting with the environment, communicating wirelessly and perform limited processing. In every time step, the sensor nodes need to take decision about some tasks to perform. Our proposed algorithm helps sensor nodes to learn the usefulness of each task based on reinforcement learning. We present an object tracking application with online scheduling of tasks based on our proposed approach. Our simulation studies show a more efficient task scheduling than traditional resource management schemes such as static scheduling, random scheduling and independent reinforcement learning based scheduling of tasks.**

Keywords- Cooperative reinforcement learning; Resource Coordination, Tasks scheduling; Wireless sensor networks.

I. INTRODUCTION

The effective coordination of the available resources is a very important aspect in resource limited systems such as wireless sensor networks. In general resource coordination means to regulate or manage the resources in such a way that the planned tasks can be achieved effectively. There are various resources of interest in wireless sensor networks such as energy, computing performance, memory, available information or data, communication capabilities and processing functionality.

As a WSN is very much application specific and resource constrained, careful consideration should be given to resource coordination. Object tracking, area monitoring, coverage, clustering, routing and in-network data aggregation are some of the common applications in sensor networks. For these applications sensor nodes need to perform some tasks like sensing, transmitting, receiving, sampling, sleeping etc. Due to resource scarcity it is very much needed to coordinate the resources

Resource coordination allows sensor nodes to self schedule its tasks with only local information. It helps to find out the best allocation of tasks to resources for the specific application by performing some triggering activities such as by applying some learning algorithms. It also assists to learn usefulness of tasks in any given state to achieve the most resource effective implementation. This triggering can be performed by several ways such as offline, periodic, on demand or can be issued by changes in network.

For example, if we consider some sensor nodes scattered in a particular area for object tracking application, we need to model the network in such a way that it can efficiently allocate its tasks to resources. For object tracking the tasks needed for the sensor nodes are sensing, transmitting, receiving, aggregating and sleeping. If we apply resource coordination in this system for object tracking, we need to model this in such a way that it can maximize the network lifetime over time. That means resource coordination maintains the quality of service by best allocation of self scheduled tasks to resources with adaptive support in dynamic environment. The scheduling of these tasks has to be determined online due to the dynamic nature of the sensor networks. Our goal is to find out the best scheduling of tasks which is resource efficient.

We propose a reinforcement learning based action scheduling scheme for resource coordination in wireless sensor networks. Cooperative Q learning is a reinforcement learning algorithm. It helps to find out the best tasks to execute at each time step. In this learning algorithm, we have a set of states which is defined by system variables (object in the field of view of nodes, data to transmit, residual energy etc.), a set of tasks and a reward function to calculate the Q values. Tasks scheduling by this algorithm gives better performance in energy efficiency comparing with other existing tasks scheduling algorithms. It schedules the tasks in such a way that the energy consumption is optimized. Existing works do not consider any cooperation of sensor nodes for resource coordination. This work considers cooperation of nodes for resource coordination.

To our knowledge this is the first work applying cooperative reinforcement learning for resource coordination.

The rest of this paper is organized as follows. In Section 2, we briefly describe the related works. Our reinforcement learning approach for resource coordination is presented in section 3. An application using our proposed approach is mentioned in Section 4. In Section 5, we describe our simulation results and evaluation. Section 6 concludes this paper with summary.

II. RELATED WORKS

In distributed networks, the goal is to build a collaborative environment for facilitating the effective usage of resources [11]. Resource coordination research addresses this issue and aims at creating an environment where nodes are able to manage the different resources. They can cooperate to provide value-added services such as to increase the lifetime of the network, which could not be provided if the nodes were to operate individually. To find out the best scheduling of tasks in a collaborative environment is a challenging issue for resource coordination. Only a few related works are available which considers tasks scheduling for resource coordination. Most of the existing methods do not provide online scheduling of tasks. There is no work which considers the cooperation of nodes for resource coordination.

K. Shah et al. [1] proposed independent reinforcement learning approach for resource management in wireless sensor networks. In their approach, no cooperation between nodes for the resource coordination is considered. In [8], a predefined static action scheduling approach is proposed. T. Liu et al. [9] proposed random selection of tasks for scheduling. We have compared our approach with these existing approaches. It results in a better performance comparing to these in terms of cumulative reward over time and also in residual energy of the network.

III. COOPERATIVE Q LEARNING FOR RESOURCE COORDINATION

Cooperative Q learning is a reinforcement learning approach to learn the usefulness of some tasks over time in a particular environment. Reinforcement learning is a branch of machine learning and is concerned with determining an optimal policy. It maps states of the world to the actions that an agent should take in those states so as to maximize a numerical reward over time [10].

We consider the wireless sensor network as a multi-agent system. The nodes correspond to agents in the multi-agent reinforcement learning. The world surrounding the sensor nodes forms the environment. Tasks are considered as activities for the sensor nodes at each time step such as transmit, receive, sleep, sense etc. States are formed by set of system variables such as object in the field of view of sensor nodes, required energy for a specific action, data to transmit etc. A reward function provides some positive or negative feedback for performing a task at each time step. Value functions define what is good for an agent over long run described by reward function and some parameters.

In cooperative Q learning every agent needs to maintain a Q matrix for the value functions like independent Q learning. Initially all entries of the Q matrix are zero and the nodes or agents may be in any state. Based on the application defined variable or system variables, the system goes to a particular state. Then it performs an action which depends on the status of the nodes (Example: For transmit action, a node must have residual energy which is greater than transmission cost). It calculates the Q value for this (state, task) pair with the immediate reward.

$$Q^i_{t+1}\left(s^i_t, a^i_t\right) = (1-\alpha) Q^i_t\left(s^i_t, a^i_t\right) + \alpha\left(r^i_{t+1}(s^i_{t+1}) + \gamma \sum_{j \in Neigh(i)} f^i(j) V^j_t(s^j_{t+1})\right)$$

$$V^i_{t+1}(S^i_t) = \max_{a \in A^i} Q^i_{t+1}(S^i_t, a)$$

Where, $Q^i_{t+1}\left(s^i_t, a^i_t\right)$ means the updates Q value for agent *i* at time *t+1*, after executing the action *a* at time step *t*.

r^i_{t+1} means the immediate reward after executing the action *a* at time *t*.

V^j_t is the value function for agent *j* at time *t.*

V^i_{t+1} is the value function for agent *i* at time *t+1*.

$\max_{a \in A^i} Q^i_{t+1}(S^i_t, a)$ means the maximum Q value after performing an action from the action set A for the agent *i.*

γ is the discounted factor which can be set to a value in [0,1]. The higher the value, the greater the agent relies on future reward than the immediate reward.

α is the learning rate parameter which can be set to a value in [0,1]. It controls the rate at which an agent tries to learn by giving more (close to 1) or less (close to 0) weight to the previously learned utility value.

$f^i(j)$ defines the factor that weights the value functions of the neighbours of agent *i* in the computation of its own value function. We define it as follows:

$$f^i(j) = \begin{cases} \dfrac{1}{|Neigh(i)|}, Neigh(i) \neq 0 \\ otherwise \\ f^i(j) = 1 \end{cases}$$

The algorithm can be stated as follows:

Initialize the values Q(s,t) to zero. Where **s** is the set of states and **t** is the set of tasks.

Run until the residual energy becomes zero:

- Start from current state s defined by the variable.

- From current state, calculate the Q values for all actions in the action sets with the reward function.
- Calculate the value function for the task which gives maximum Q value.
- Send the value function to the neighbours.
- Node goes to next state based on the performed task which has the highest Q value.

IV. OBJECT TRACKING APPLICATION USING OUR APPROACH

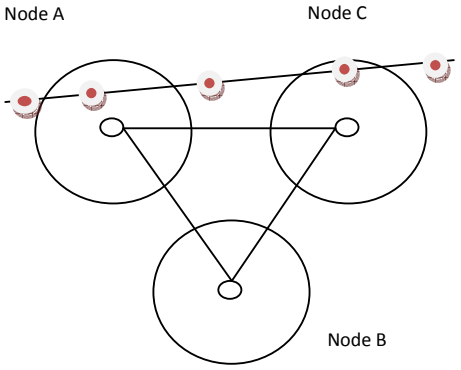

Figure 1. Object tracking example.

Figure 1 shows the movement of a single object, ie. the "red dotted circles" represents the object's position at different time points. Consider three nodes, which are fully connected. Each node has no information of what actions are better for them in terms of energy consumption. They will learn by performing some actions over time based on their utility.

Consider a snapshot of the system that the object is within the field of view of Node A and is moving and finally entering the field of view of Node C.

Nodes A, B and C have no idea about what tasks are better for them. Suppose Node A is in a particular state based on application defined or system variables. In that particular state, the node tries out all the actions and finds out a particular one which maximizes the Q value. Then it moves to the next state after performing that action. Node A sends the value function to its neighbours. If there is something to transmit then after transmission, it gets a positive reward otherwise a negative. Going to that state, it selects another action which has maximum Q value. Now in this example, when an object enters into the field of view of Node A, it receives a positive reward after executing sensing task. So, Node A learns after some times that it receives a positive reward for sensing if something in its field of view. The object has moved away from the sensing range of node A and does not exist in the range of any nodes. In this case, all nodes will benefit from staying in the sleeping mode. The object moves in to the field of view of node C and it receives a larger reward for sensing. After that it will move to another state by calculating the Q value. Node B has nothing to transmit or receive. So, it will benefit by remain silent at this particular state of the environment.

In this way, each node is able to decide on the "best" available tasks for a particular state in a dynamic environment.

V. SIMULATION RESULTS AND EVALUATION

We have performed two different experiments for finding the efficiency of our proposed approach. In both experiments, we have considered the object tracking application. States are defined by system variables. In the first experiment, we have considered three variables which are "object in field of view (FOV)", "data to transmit (DTT)" and "residual resource level (RL)".

In the second experiment, we have considered some variables which correspond to "number of genrated packets", the "timer of completing delivery of a packet", the "residual energy of the node", the "transmit cost", the "receive cost", the "sense cost", the "sleep cost" and the "number of connections to the neighbour nodes". Two different sets of state space help to find out the efficiency of tasks scheduling. We have considered sleep, sense, transmit and receive as our tasks set for our both experiments.

A. First Experiment

In first experiment, we have considered a setup of nodes like in figure 1. We have our simulation environment using Java. We have considered the following set of states:

1. FOV=0 and RL>= Minimum Energy required for Transmission and DTT= 0
2. FOV=0 and RL >= Minimum Energy required for Transmission and DTT=1
3. FOV=0 and RL < Minimum Energy required for Transmission and DTT=1
4. FOV=1 and RL>= Minimum Energy required for Sensing and DTT= 0
5. FOV=1 and RL < Minimum Energy required for Sensing and DTT= 0
6. FOV=1 and RL<Minimum Energy required for Sensing and DTT=1
7. FOV=1 and RL>= Minimum Energy required for Sensing and DTT=1
8. FOV=0 and RL>= Minimum Energy required for RL< Minimum Energy required for Transmission and DTT= 0

Suppose if the variable, object in field of view is equal to 1 that means there is object in the sensing range of that node. In each state, the node tries out the tasks from task sets and finds out the task which has higher Q value. Task

will change the state variables. For example, after performing sense task in state 1, resource level will be equal to (residual energy minus energy required for sensing) and field of view will be equal to 1. Now to calculate the Q values we need reward function. In this experiment, we have considered the following reward function. We have assigned the most reward to the task which needs the least amount of energy. Here is the ascending order of tasks for energy consumption: Sleeping, Sensing, Receiving and Transmitting.

TABLE I. REWARD FUNCTION FOR EXPERIMENT 1

Tasks	Reward
Sleeping	5
Sensing	4
Receiving	3 for Success
	-3 for Not Success
Transmitting	2 for Success
	-2 for Not Success
γ	0.5
α	0.5

Table 1 shows the reward function that we have considered for experiment 1.

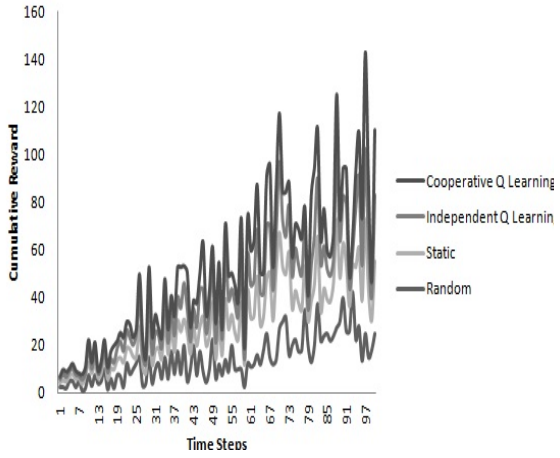

Figure 2. Cumulative reward over time.

We present a comparative performance analysis of our approach against three other methods: 1. Static: where each node performs a fixed set of actions repeatedly without any learning about the best scheduled tasks for optimized energy consumption. 2. Random: where each node performs an action randomly chosen from uniform distribution each time and 3. Independent Q Learning [1]: where each node independently takes decision about the tasks by simple reinforcement learning.

In figure 2, we show that the cumulative reward over time for our approach is greater than for the other approaches. Higher cumulative reward means that the node has chosen tasks in such a way that it maximized the reward. As our reward is defined to minimize the energy consumption, we will get less energy consumption for tasks by getting higher cumulative reward.

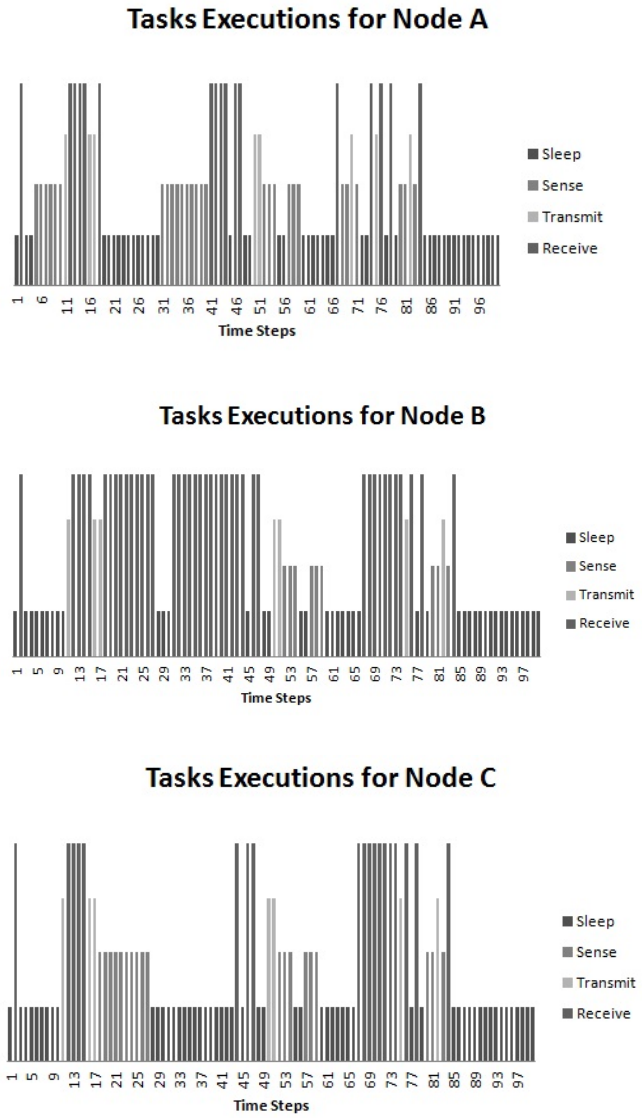

Figure 3. Task executions for Node A, B and C

In figure 3, we show the task scheduling of node A, B and C considering the case of figure 1 with cooperative Q learning. Each node has no information of what tasks are better for them and try to learn over time based on the utility of their tasks. For example, node A does not know that it needs to sense as object is in its field of view. Here each bar represents a task executed at each time step. We represent "Receive", "Transmit", "Sense" and "Sleep" tasks in descending order of height of the bars. We can observe that

node A immediately learns that it is getting paid to sense as object is in its field of view. In the middle of the simulation time, as the target is out of reach of all sensor nodes, all nodes will be rewarded for sleeping. Similarly, we can observe that for node C, it will be rewarded for sensing after some times when the object is in its field of view. After that when the object is out of reach of all sensor nodes, all the nodes will be rewarded for sleeping.

In figure 4, we show the number of execution of each task for each method. We have performed this experiment for all methods by considering the case of figure 1. Here we can see that in our approach the number of execution of sleep task is larger than other methods. As sleeping requires less energy among all tasks, our appraoch shows more energy efficiency by performing more sleep action over time.

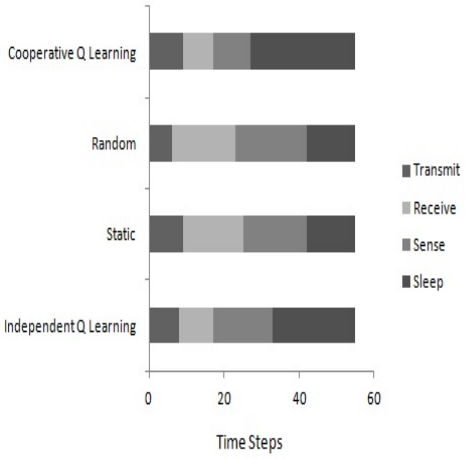

Figure 4. Total number of execution for each action.

B. *Second Experiment*

In our second experiment we consider a different set of states, the same tasks set (sleep, sense, transmit, receive) and different reward function. This state space considers more state variables. We consider some parameters which designed our state space. It's a multi object tracking systems. There are some moving objects throughout the system. Each node is connected with neighbours like a fully connected graph. If a particle enters in to the sensor radius of a particular node, it generates a packet.

For the number of generated packets, we have used "sNodes.aPackets.Count".

For the timer of completing delivery of a packet, we have used "iTransmitting".

For the residual renergy of the node, we have used "SNodes.iResidualEnergy".

For sensor cost, transmit cost, receive cost and sleep cost we have used "iSensorCost", "iTransmitCost", "iReceiveCost" and " iSleepCost" respectively.

The state space can be defined as follows:

- sNodes.aPackets.Count>0 and sNodes.iResidualEnergy>= iSensorCost

This condition may be True or False.

- sNodes.aPackets.Count>0 and iTansmitting=0 and sNodes.iResidualEnergy<iTransmitcost

This condition may be True or False.

- sNodes.aconnections!=NULL and sNodes.iResidualEnergy<iRecievecost

This condition may be True or False.

TABLE II. REWARD FUNCTION FOR EXPERIMENT 2

Tasks	Reward
Sleeping	0.05
Sensing	0.001
Receiving	(0.3-iReceiveCost) for Success
	(-iReceiveCost) for Not Success
Transmitting	(0.2-iTransmitCost) for Success
	(-iTransmitCost) for Not Success
γ	0.5
α	0.5

Considering these reward functions, state and action sets, we have simulated five sensor nodes in this environment. We have also compared our approach against other approaches for cumulative reward over time.

Figure 5 shows the comparison graph of our approach with the other approaches. It shows a better cumulative reward over time for our approach.

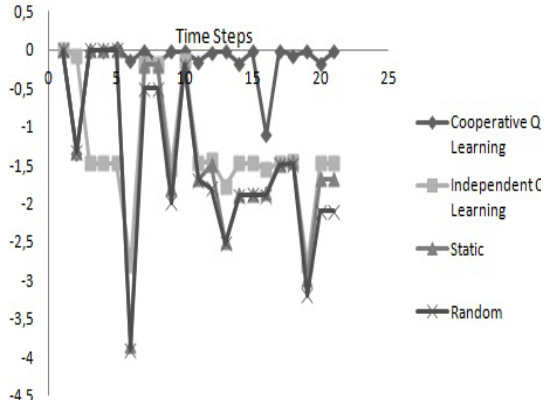

Figure 5. Cumulative reward over time in experiment 2.

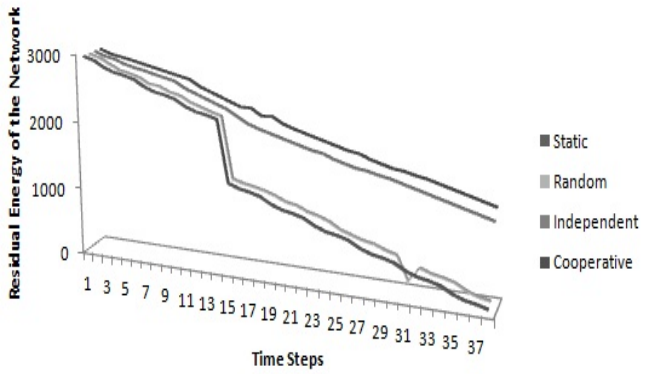

Figure 6. Residual energy of the network over time

Figure 6 shows the comparison graph for the residual energy of the network. Considering the target movement as in figure 1, we calculate the residual energy of the network at each time step. We assume energy budget for each node as 1000 unit. For sleeping we spend 10 unit, for sensing 15 unit, for receiving 20 unit and for transmitting 30 unit of energy. We have found better result for cooperative Q learning comparing with other approaches. So, it helps to increase the lifetime of the network.

From the simulation result and evaluation, we can say that by applying cooperative Q learning it is possible to learn the usefulness of actions to perform in different states. Cooperative Q learning scheduled the tasks in such a way that the energy consumption is reduced. It also receives better cumulative reward over time comparing with other approaches.

VI. CONCLUSION

Effective resource coordination is an important issue in WSN. To increase the lifetime of the network and to overcome the problem of resource deficit, resource coordination is very necessary. Tasks scheduling is an effective way for resource coordination. We have used a reinforcement learning approach for tasks scheduling. Our simulation results show that our approach gives better scheduling comparing with other methods. It also improves the cumulative reward over time. Higher cumulative reward means it consumes less energy by performing scheduled tasks. It also gives better performance for residual energy of the network which helps to increase the lifetime of the network.

The paper is based on our initial work to apply reinforcement learning for action scheduling in sensor networks. We have not considered so many states and tasks for our work. To design our state space with more dynamic variables of the environment, to take the consideration of processing as an action is one aspect of our future work. We have performed our experiments on a fully connected topology. To apply our learning algorithm in random deployment of nodes with different topologies can be another future work. To apply other learning algorithms and to find out the efficiency comparing with our proposed approach can extend our future work.

ACKNOWLEDGMENT

This work was supported in part by the Erasmus Mundus Joint Doctorate in Interactive and Cognitive Environments, which is funded by the EACEA Agency of the European Commission under EMJD ICE FPA n° 2010-0012.

REFERENCES

[1] K.Shah and M. Kumar, "Distributed Independent Reinforcement Learning (DIRL) Approach to Resource Management in Wireless Sensor Networks," Proc. IEEE Mobile Adhoc and Sensor Systems (Mass 07), IEEE Press,2007, pp.1-9, doi: 10.1109/MOBHOC.2007.4428658.

[2] R. Sutton and A. Barto, Reinforcement Learning: An Itroduction, MIT Press, ISBN 9-262-19398-1.

[3] Y.Shoham, R.Powers, and T.Grenager, "Multi Agent Reinforcement Learning: A critical Survey," Computer Science Dept., Stanford University,California, US, Tech. Rep., 16 May 2003.

[4] K. Tuyls and A. Nowe, "Evolutionary Game Theory and Multi Agent Reinforcement Learning," The Knowledge Engineering Review, vol.20, no.1, pp. 63-90,2005.

[5] G. Chalkiadakis, Multiagent reinforcement learning: Stochastic games with multiple learning players, Dept. of Computer Science, University of Toronto, Canada, Tech. Rep., 25 March 2003, URL: http://www.cs.toronto.edu/.gehalk/DepthReport/DepthReport.ps.

[6] M. Bowling, Multiagent learning in the presence of agents with limitations, Ph.D. dissertation, Computer Science Dept., Carnegie Mellon University, Pittsburgh, US, May 2003.

[7] L. Panait and S. Luke, Cooperative multi-agent learning: The state of the art,. Autonomous Agents and Multi-Agent Systems, vol. 11, no. 3, pp. 387-434, November 2005.

[8] P.Levis and D.Culler, Mate: A Tiny virtual Machine for Sensor Networks. In proc of the 10th International Conference on Architectural Support for Programming languages and Operating Systems, Oct 2002.

[9] Tin Liu, Margaret Martonosi, Impala: A Middleware system for managing autonomic, parallel sensor systems. In ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming , 2003.

[10] V. Lesser, C. Ortiz and M. tambe, Distributed Sensor Networks: A Multi Agent Perspective, Kluwer Publishers, 2003.

[11] A. Lopes and L. Botelho, Improving Multi Agrnt Based Resource Coordination in Peer to Peer Networks, Journal of Networks, vol. 3, no. 2, February 2008.