Design of a Multi-Agent System for Hierarchical Network Management in Wireless Sensor Network

Mubashsharul I. Shafique, Haiyi Zhang and Yifei Jiang

Jodrey School of Computer Science, Acadia University
Wolfville, NS, B4P 2R6, Canada

Abstract. In order to manage a sensor network efficiently, we can divide it logically into disjoint parts, called clusters. As sensor nodes are resource-constraint, it is desirable to chose suitable cluster heads and do role-switching of the cluster heads wherever appropriate. In this work, we design a system that selects new cluster head through collaboration of multiple software agents in each cluster. Our system allows to pick up cluster heads dynamically based on current network status. Agents in our design use *Fuzzy Logic*-based controller to find new cluster heads.

1 Introduction

Wireless Sensor Network is a deployment of sensor nodes that do data reporting to interested user via sink node. It offers a great facility to remotely monitor an unattended environment. Due to this feature, sensor networks are widely used in military surveillance, habitat monitoring, industrial plants, and in many other places. Once a sensor network is deployed, it is necessary to manage it efficiently in order to optimize network life-time.

A sensor network can be managed by hierarchical or flat organization. However, as hierarchical organization has several benefits over flat network structure, normally a sensor network is divided into clusters. Here in this work, we design a multi-agent system to rotate the role of cluster head nodes. Software agents in our design autonomously collaborate with each other in the distributed sensor network environment. The remaining part of this paper is organized as follows: section 2 presents some background knowledge and section 3 defines the problem that his work addresses. Next, section 5 explains the design of our system and inter-agent communication is presented in section 6. We conclude our paper in section 7.

2 Background Knowledge

2.1 Wireless Sensor Network

Wireless Sensor Network(WSN) consists of autonomous sensor devices (known as nodes or motes) that can sense an environment. For example, MicaZ sensor nodes from

Shafique M., Zhang H. and Jiang Y. (2010).
Design of a Multi-Agent System for Hierarchical Network Management inWireless Sensor Network.
In *Proceedings of the 6th International Workshop on Artificial Neural Networks and Intelligent Information Processing*, pages 53-62

Crossbow[1] can sense ambient light, barometric pressure, GPS, magnetic field, sound, photo-sensitive light, photo resistor, humidity and temperature.

Often times, these sensor nodes are battery-powered, and equipped with limited processing capacity and memory. However, WSNs are very effective and efficient at monitoring remote environment and communicate real-time data. Sensor nodes can detect events or phenomena, collect and process data, and transmit sensed information to the interested users.

2.2 Virtual Organization in Sensor Network

WSN differs from an IP network in regards to network backbone because, often times a WSN does not have any fixed infrastructure. Sensor nodes may run out of battery power or network topology may change due to some mobile nodes. So, in order to support data routing, WSN relies on virtual infrastructure which forms on-the-fly during system operation. One way to manage a sensor network, is to logically divide it into some disjoint clusters. A particular node in each cluster works as cluster head, and gathers data from other nodes of the cluster. The cluster head then forwards data to the next hop node towards the data sink, which is the final destination of any data. Figure 1 shows an example sensor network with three clusters. We show data reporting from a normal sensor node to the sink by arrow heads.

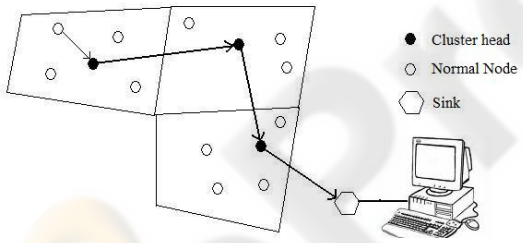


Fig. 1. Virtual organization in WSN.

3 Problem Definition

In a static network topology, cluster heads of a sensor network die quite early due to excessive relaying of data stream towards sink. So, in order to prolong network lifetime, a network should rotate the role of cluster heads. For example, if a current cluster head had drained a significant amount of energy, network should keep the provision to pick a new cluster head for that cluster. This kind of dynamic load balancing can delay the first node death and help decrease data packet loss in the network thereby.

Due to the large number of sensor nodes in a typical sensor network, it is not realistic to run any centralized algorithm to switch cluster heads. A network needs to have distributed architecture which can select new cluster heads with local information, and further, the role-switching should remain quasi-transparent to the remaining network.

[1] http://www.xbow.com/.

4 Related Works

For Wireless sensor Networks (WSNs), the major design goal is to minimize energy consumption and maximize the network lifetime. In the last few years, plenty of attempts of exploring advanced power conservation approaches have been used by researchers for wireless sensor networks. Cluster-based routing approach is one of the famous energy efficient routing approaches in WSNs. LEACH (Low Energy Adaptive Clustering Hierarchy) in [1], is the first hierarchical cluster-based routing protocol for WSNs. This algorithm uses random and periodic rotation of the Cluster Heads (CHs) for load balancing, which can evenly disperse the energy load among the sensor nodes in the network. More specifically, the cluster heads belonged to the corresponding clusters will only use for certain number of rounds. After a predefined round, new cluster heads will be randomly generated, which is based on a role that if the random number is less than a calculated threshold $T(n)$, the corresponding sensor node will be selected as the cluster-head for the current round. This randomized periodic role rotation ensures that all the nodes are equally likely to be cluster head nodes. However, simply random cluster head rotation will select unfavorable cluster heads, which will turn out high energy consumption in later rounds.

Due to the above reason, in [9], Energy-LEACH protocol improves the procedure of CH rotation. Similar to LEACH protocol, the process of CH rotation of E-LEACH is divided into rounds. In the first round, each sensor node has the same probability to be turned into CH, which means sensor nodes are randomly selected as CHs. In the next rounds, since the residual energy of each sensor node is different after every communication round, residual energy of node is considered as the main metric that decides whether the sensor nodes turn into CHs or not after the first round. The sensor nodes who have more remaining energy will become CHs rather than the ones with less remaining energy.

As for the aforementioned two LEACH algorithms, there is a shared drawback, that is, both of them consumes more CPU cycles, since each sensor node in WSNs has to calculate the threshold and generate the random numbers in each round. To overcome this shortcoming, a improved LEACH, called LEACH-C protocol, is proposed in [10]. LEACH-C is a centralized clustering algorithm and uses the same steady-state phase as in LEACH algorithm. This protocol also produces better performance on cluster heads rotation. During the set-up phase of LEACH-C, each sensor node sends information about its current location (this may determine by using GPS) and residual energy level to the Base station (BS). In order to ensure that the energy load is evenly distributed among all the sensor nodes in WSNs, The average node energy is computed by the BS, and then, determines which sensor nodes have energy below this average. Once the CHs and associated clusters are found, the BS broadcasts a message that obtains the cluster head ID for each node. If a cluster head ID matches its own ID, the node will be selected as a cluster head; otherwise the node determines its TDMA slot for data transmission and goes sleep until it's time to transmit data. The steady-state phase of LEACH-C is identical to that of the LEACH protocol.

Based on the above approach, in [11], cluster heads election using fuzzy logic is proposed, which can minimize energy consumption and provide a substantial increase in network lifetime compared with the probabilistically cluster heads selecting

approaches. According to this proposed approach, for a cluster, the node elected by the base station is the node having the maximum chance to become the cluster-head, which is based on three fuzzy descriptors: energy level in each sensor node, sensor node concentration and node centrality with respect to the entire cluster. The operation of this fuzzy logic cluster-head election scheme is divided into two rounds with each consisting of a setup and steady state phase similar to LEACH algorithm. During the setup phase, fuzzy knowledge processing is used for determining the CHs, and then the cluster is organized. In the steady state phase, the aggregated data is collected by CHs. After that, CHs perform signal processing functions to compress the data into a single signal. This composite signal is then sent to the base station. Fuzzy logic control model is core part of this proposed approach; it includes a fuzzifier, fuzzy rules, fuzzy inference engine, and a defuzzifier. To be specific, fuzzifier is used to take the crisp inputs from each of variables of energy, concentration and centrality and determine the degree to which these inputs belong to each of the appropriate fuzzy sets. Then, these fuzzified inputs are applied to the antecedents of the fuzzy rules. As for the defuzzification, the input for this process is the aggregate output fuzzy set chance and the output is a single crisp number. For fuzzy inference engine, Mamdani Method [13] is commonly used. In [12], similar cluster heads selections based on fuzzy logic algorithm are also presented.

To our knowledge, several attempts have also been used by some researchers to reduce energy consumption based on mobile agents [2][3][4][5]. Due to the constraints of bandwidth in wireless sensor network, the network's capacity may not satisfy the transmission of sensory data. In order to handle the problem of overwhelming data traffic, Qi, et al. [6] proposed Mobile Agent-based Distributed Sensor Network (MADSN) for multi-sensor data fusion. For this proposed approach, it not only achieves data fusion, but also reduces energy expenditure. However, the application of this approach can only be applied on cluster-based topologies. MADD approach in [7] is introduced to deal with this problem. Currently, most energy-efficient proposed approaches are focused on data-centric model, such as the directed diffusion. By selecting good path to drain quality data from source nodes, directed diffusion approach can achieve substantial energy gain. However, it still allows redundant sensory traffic to flow back to the Base station. The main advantage of MADD is to reduce the redundant sensory data. Through using mobile agent, data is aggregated at each source node and is brought back to sink. This allows substantial energy gain toward the network lifetime.

To explain the process of MADD approach, it starts when the mobile agent is dispatched from the BS with the interest and ends when it returns to the sink with the aggregated data. The processes involved in MADD are divided into three phases. First, the mobile agent is dispatched from BS to the first source node. Second, the mobile agent shifts from the first source node to the last source node, visiting selected source nodes in between. The drawback for this approach is that it doesn't always guarantee the best sequence of nodes to be visited.

To deal with the aforementioned limitations, Shakshuki et al. in [8] proposed a mobile agent for efficient routing approach (MAER) by using both Dijkstra's algorithm and Genetic Algorithms (GAs). As we know, the order of source nodes to be visited by the mobile agent greatly affects the energy consumption. Although MADD, the work presented in [7], allows the agent to autonomously select visit sequence of source nodes

for achieving data aggregation, it does not always provide an optimal sequence. To address this shortcoming, MAER in [8] introduces Genetic Algorithm (GA) to produce an optimal route.

5 Agent Architecture

An agent in our design consists of four modules: Problem Solver, Knowledge-base Updater, Scheduler, and Communicator. Further, each agent maintains own knowledge base in its memory, which gets updated as a result of inter-agent message communication. Here in this section, we describe different components of individual agent in detail.

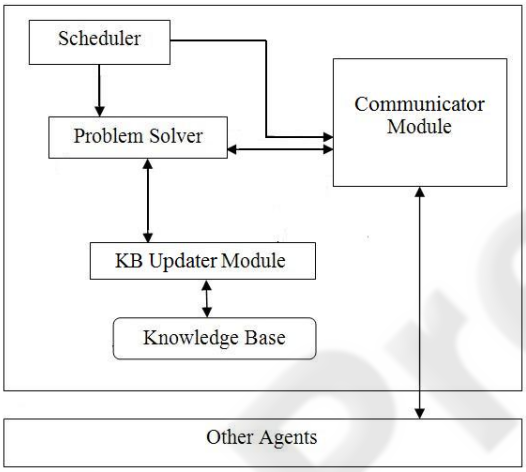


Fig. 2. Agent Architecture.

In our design, the role of an agent software switches between two modes- *Cluster head Mode* and *Normal Mode*. However, at any particular time, within each cluster, agent software of only one sensor node works in *Cluster head Mode*, others execute in *Normal Mode*.

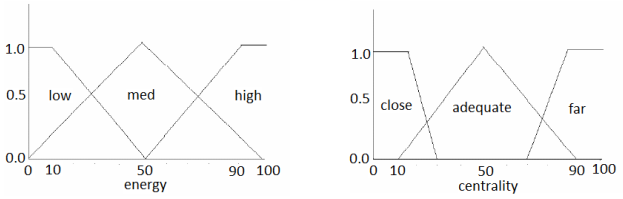
5.1 Scheduler Module (SM)

This module is in charge of scheduling a new round of cluster head selection. It is active in *Cluster head Mode* only. It interacts with *Problem Solver Module* and *Communicator Module*. After periodic interval, it checks the remaining energy of the cluster head node and initiates new cluster head selection if necessary.

5.2 Problem Solver Module (PSM)

This is the basic working module of an agent. Based on current working mode, PSM does two distinct tasks:

Weight Calculation in Normal Mode. We define the weight of a sensor node as a function of two parameters: *Centrality(C)* and *Remaining Energy(RE)*. As sensor nodes are resource-constraint, we use simple Fuzzy Logic based controller in agents to calculate node weight. During *Fuzzification* phase, we followed the mapping presented in [11] to convert crisp values to fuzzy values. Figure 3 portrays this *Fuzzification* process. We define our own rule-base for fuzzy controller in table 1. PSM uses this rule base to determine current weight value in fuzzy variable.



(a) Fuzzy set for fuzzy variable *energy* (b) Fuzzy set for fuzzy variable *centrality*

Fig. 3. Fuzzy Sets used in Fuzzy Controller.

Table 1. Fuzzy Rule Base.

Rule	Centrality	Energy	Result
1	Close	High	High
2	Adequate	High	High
3	Far	High	High
4	Close	Medium	High
5	Adequate	Medium	Medium
6	Far	Medium	Medium
7	Close	Low	Medium
8	Adequate	Low	Low
9	Far	Low	Low

New Cluster Head Selection in Cluster head Mode. PSM of cluster head agent operates on all Fuzzy weight values of normal sensor nodes. After comparing multiple Fuzzy weight values, if some nodes are equally suitable to be the new cluster head, PSM runs following scheme to break a tie:

In order to *Defuzzify*, we assign numeric values against different Fuzzy values *High/Close*=3, *Medium/Adequate*=2, and *Low/Far*=1 and bias the decision by putting 60% importance to *energy*. For example, if two nodes n_i and n_j are in a tie with Fuzzy values {Far, Medium} and {Close, Low} respectively, then PSM computes *Defuzzified* values as d_i = 1*0.4 + 2*0.6 = 1.6 and d_j = 3*0.4 + 1*0.6 = 1.8. Here d_i and d_j are crisp values for node n_i and n_j respectively. Even after this computation, if PSM can not break the tie, it compares node IDs and selects node with minimum ID as next cluster head.

5.3 Communicator Module (CM)

This module is in charge of sensor radio component and communication with other agents in the system. If agent software is running in *Cluster head Mode*, it does two things: (a) Upon receiving request from SM, it sends out *Discovery Message* to other agents; (b) If it receives *Weight Message*(s), it forwards weight value(s) to PSM.

On the other hand, if the agent is working in *Normal Mode*, CM does only one thing: it receives *Weight Message*(s) and forwards to the *Knowledge-base Updater Module*.

5.4 Knowledge-base Updater Module (KBUM)

This is the only module that interacts with agent *Knowledge-base*. KBUM uses a dedicated area of sensor node's memory to maintain the *Knowledge-base*. Any change in shared knowledge like- weight updates or rule updates are propagated to it by PSM and/or CM; and as a result, KBUM synchronizes agent's local memory with global state.

5.5 Knowledge-base (KB)

Knowledge-base in an agent holds a local copy of the shared knowledge. An agent's KB houses a snapshot of a fragment of global information, that it is interested in. KB includes rules for *Fuzzy* controller, latest weights of all neighbor nodes and current cluster head's node ID.

6 System Dynamics

6.1 Protocol Message Types

In a collaborative multi-agent system, message passing is an integral part which facilitates distributed processing. In this section, we present four different message types that agents in our design exchange during system operation.

Discovery Message. Once remaining energy of the current cluster head goes below a threshold value (which is 40% of initial energy in our implementation), cluster head agent broadcasts *Discovery Message*. A *Discovery Message* initiates new cluster head selection within a cluster.

Weight Message. *Weight Message* contains latest weight value of a node. This message is broadcasted by agent in a normal node as a response to an incoming *Discovery Message* from current cluster head.

Control Message. *Control Message* is used to broadcast any change in routing topology. After determining new cluster head, current cluster head agent broadcasts *Control Message* indicating new cluster head node ID.

Data Message. *Data Message* originates from normal node agents and is used to report current data readings to the cluster head.

6.2 System Work-flow

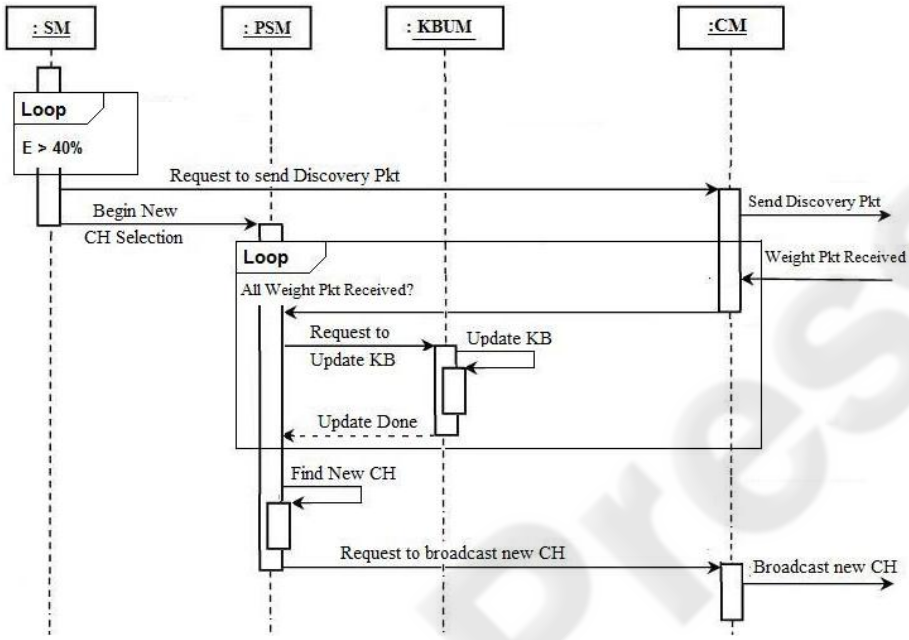


Fig. 4. Sequence diagram for *cluster head mode*.

In this section we explain the sequences of operation in our agent-based system. First, figure 4 shows action sequences for an agent operating in *Cluster head Mode*. After periodic interval, SM measures remaining energy of the cluster head node, and if it is bellow the threshold (*i.e.*, less than 40% of initial energy), SM starts a new cluster head selection round. It notifies PSM that a new cluster head selection process is in effect and also requests CM to send out *Discovery Message*.

CM then sends out *Discovery Message* in the wireless medium, and in response, it receives *Weight Message*s from other agents. CM forwards these node-weights to PSM for calculation. In turn, PSM requests KBUM to update *Knowledge base* with newly received weight values and once PSM receives weights from all nodes, it uses *Rule base* to select new cluster head as explained in section 5.2. After finding the new cluster head, PSM requests CM to convey other nodes about new cluster head, and as a result, CM sends out new *Control Message* containing newly chosen cluster head ID.

Figure 5 portrays action sequence of an agent working in *Normal Mode*. If its CM receives a *Discovery Message* from current cluster head, CM requests PSM to compute

own node weight. Upon computation, PSM returns own weight to CM. CM then encapsulates this weight value in a *Weight Message* and broadcasts that message in the neighborhood. Besides, if CM receives *Weight Message* from a different node, it requests KBUM to update agent's *Knowledge base*. In a similar way, KB update is also performed if an agent receives *Control Message* from current cluster head.

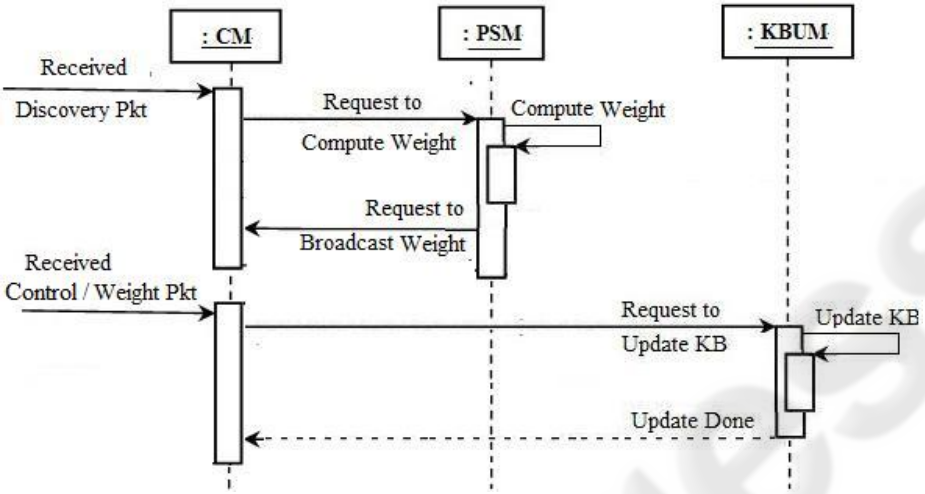


Fig. 5. Sequence diagram for *normal mode*.

7 Conclusions

In this work, we design a system to rotate the role of cluster heads in a Wireless Sensor Network. Multiple agents, each residing in individual sensor node, interact with each other and participate in new cluster head selection. In our design, agent-based architecture provides flexibility to network management. At the same time, as we use fuzzy controller, this design can be easily extended by adding new rules in the rule-base. Our future work focuses on implementing this design in TinyOS operating system for Wireless Sensor nodes.

References

1. W. Heinzelman, A. Chandrakasan and H. Balakrishnan: Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In: 33rd Hawaii International Conference on System Science (2000)
2. Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang: Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. Computer Journal, vol. 47, no. 4, pp. 448-460, (2004)

3. H. Qi, S.S. Iyengar, and K. Chakrabarty: Multi-Resolution Data Integration Using Mobile Agents in Distributed Sensor Networks. IEEE Trans. Systems, Man, and Cybernetics Part C: Applications and Rev., vol. 31, no. 3, pp. 383-391, Aug. (2001)
4. Daniel Massaguer, Chien-Liang Fok, Nalini Venkatasubramanian, Gruia-Catalin Roman, Chenyang Lu: Exploring sensor networks using mobile agents. pp. 323-325, AAMAS (2006)
5. C.-L. Fok, G.-C. Roman, and C. Lu: Mobile agent middleware for sensor networks: An application case study. In: 4th Int. Conf. on Information Processing in Sensor Networks (IPSN'05), pages 382–387. IEEE, April (2005)
6. Hairong Qi, Yingyue Xu, Xiaoling Wang: Mobile-agent-based Collaborative Signal and Information Processing in Sensor Networks. In: Proceeding of the IEEE, Vol. 91, NO. 8, pp.1172-1183, Aug (2003)
7. Min Chen, Taekyoung Kwon, Yong Yuan, Yanghee Choi and Victor C. M. Leung1: Mobile Agent-Based Directed Diffusion in wireless Sensor Networks. EURASIP Journal on Advances in Signal Processing, Article ID 36871, (2007)
8. Shakshuki, E., Xing, X.Y. and Malik, H: Mobile Agent for Efficient Routing among Source Nodes in Wireless Sensor Networks. In: 3rd International Conference on Autonomic and Autonomous Systems (ICAS'07), June(2007)
9. Xiangning Fan and Yulin Song: Improvement on LEACH Protocol of Wireless Sensor Network. International Conference on Sensor Technologies and Applications (SENSORCOMM 2007), pp.260-264, (2007)
10. W. Heinzelman, A. Chandrakasan and H. Balakrishnan: An Application-Specific Protocol Architecture for Wireless Microsensor Networks. IEEE Transactions on Wireless Communications, Vol. 1, No. 4, October(2002)
11. Indranil Gupta, Denis Riordan and Srinivas Sampalli: Cluster-head Election using Fuzzy Logic for Wireless Sensor Networks. In: 3rd Annual Communication Networks and Services Research Conference, pp.255 - 260, (2005)
12. Xiaorong Zhu and Lianfeng Shen: Near optimal cluster-head selection for wireless sensor networks. Journal of Electronics (China), Science Press, co-published with Springer-Verlag GmbH, Vol. 4, pp.721-725, November(2007)
13. M. Negnevitsky: Artificial intelligence: A guide to intelligent systems. Addison-Wesley, Reading, MA (2001)