Exploring Sensor Networks using Mobile Agents

Daniel Massaguer[†], Chien-Liang Fok[‡], Nalini Venkatasubramanian[†], Gruia-Catalin Roman[‡], and Chenyang Lu[‡]

[†]Donald Bren School of Information and Computer Science
University of California, Irvine
Irvine, CA 92697, USA
{dmassagu, nalini}@uci.edu

[‡]Department of Computer Science and Engineering
Washington University in Saint Louis
Saint Louis, MO, 63130, USA
{liang, roman, lu}@cse.wustl.edu

ABSTRACT

Wireless sensor networks are often difficult to program and unable to adapt to a changing environment. Mobile agent middleware promises to address both concerns by providing higher-level programming abstractions and the ability to inject new agents into a preexisting network. The unique characteristics of wireless sensor networks like resource scarcity and emphasis on spatial locality require new algorithms for controlling agent behavior. This paper presents a procedure for one specific behavior: network exploration. Network exploration is needed by many tasks ranging from simple data collection to network health monitoring. Our proposed procedure uses a genetic algorithm to determine the number of agents and their itineraries, followed by techniques for in-network adaptation to unpredictable situations like node failure. This paper presents a genetic algorithm and its adaptation strategies. The procedure is evaluated using a wireless sensor network consisting of 25 Mica2 motes running Agilla, a mobile agent middleware for wireless sensor networks.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence. Distributed Artificial Intelligence[Intelligent agents, Multiagent systems]

General Terms

Design, Algorithms, Experimentation

Keywords

Applications of autonomous agents and multi-agent systems, mobile agents, (multi-)agent planning, performance evaluation of agent systems

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

1. INTRODUCTION

Wireless sensor networks (WSNs) have attracted tremendous interest in recent years as embedded systems, microsensors, and batteries improve. A WSN consists of numerous sensors (motes) embedded within the environment that autonomously form wireless ad hoc networks. By increasing the density and scale of traditional sensing systems, a WSN enables users to gain unprecedented levels of context-awareness. Current applications for WSNs include structural and habitat monitoring [6, 5] and fire fighting [2], etc.

Due to the nature of WSNs (nodes are faulty and resource-constraint) developers are confronted with a bewildering set of conflicting requirements, mostly related to software flexibility and efficiency. For example, many WSNs are deployed for long periods of time virtually guaranteeing that the user requirements will change. Unfortunately, most WSNs run statically installed software that is loaded prior to deployment, limiting network adaptivity to the tweaking of predefined parameters. Another conflicting property stems from the fact that WSNs are expected to cover a wide geographic area servicing many users that need to run different programs.

Mobile agents offer a solution to the problems mentioned above. Unlike traditional programs, mobile agents are capable of migrating across the network carrying code and state. As they migrate, they perform application-specific tasks like taking sensor readings, performing in-network data aggregation, and coordinating with other agents to achieve a common goal. This enables the development of fluid applications capable of adapting to the intricacies of WSNs (see an example of a fire tracking scenario in [2]). Mobile agents also allow WSNs to be dynamically reprogrammed by enabling users to inject new agents into the network and allowing old ones to die. Furthermore, they allow multiple applications to co-exist since multiple agents may reside on each node (for an example, see [4]).

Before mobile agents can be effectively used in a WSN, new algorithms must be created for controlling their behavior. One topic of particular interest is *network exploration*. The goal of network exploration is to visit every node and to do so efficiently in terms of time and energy. This paper addresses the specific issue of sensor network exploration using multiple mobile agents. Network exploration is used in many applications including data collection, network diagnostic and health monitoring, residual energy scanning, topology discovery, and global reprogramming. Network exploration requires that every region of the sensor network be visited at least once in a manner that is efficient in terms

of energy consumption and latency. The order and number of visitations does not matter, though redundant visitations in general decrease efficiency and should be limited. The goal of the network exploration procedure is to determine the number of agents and each of their itineraries, and the techniques for dealing with unexpected in-network circumstances, e.g., changes in topology.

2. PROBLEM DEFINITION

Using multiple mobile agents to explore a WSN offers the benefits of flexibility, load distribution, and asynchronous operation. However, since WSNs are large and resource-constrained, these mobile agents must be designed to minimize energy utilization. Furthermore, application requirements dictate that real-time guarantees be met. This section introduces the necessary notation, formulates a minimization problem, and presents a solution.

2.1 WSN Model and Assumptions

A WSN consists of numerous nodes, each containing a set of sensors, microcontroller, external memory, radio transceiver, and power source. Nodes can sense real-world phenomena, perform local computations, store information, receive data, and transmit data. Receiving and transmitting data costs more energy than performing local computations. Moreover, we assume that the WSN is connected to the rest of the world via a single base station called the *collection point*. Finally, we assume that the amount of time an agent spends processing data at each node and the time it takes to migrate is predictable. In practice, such information may be estimated based on empirical measurements (see section 4), where the measured *time per node* includes the processing, migration, and sleeping time. This knowledge is necessary to ensure that the network is explored within a certain time.

2.2 Problem Formulation

Given a sensor network, Ω, with a single collection point, the goal is to explore the entire network by visiting every region within bounded time while minimizing energy consumption. This can be done using multiple mobile agents operating autonomously and in parallel following unique routes. These routes must start and end at the collection point.

Many applications do not need every node to be visited, especially if the nodes are physically close. For example, it may not be necessary to visit two nodes separated by just one meter if the task is to determine the average temperature along a path. To account for this, we introduce a density variable d which specifies how many of the nodes within the network need to be visited for the network to be considered explored. Specifically, a network is considered explored if all nodes $m \in \Omega$ have either been visited, or have a neighbor within d hops that has been visited. Note that if $d = 0$, the exploration problem reduces to one where every node needs to be visited.

The time constraint t is specified by the maximum length of a path, expressed as the number of nodes. We do not use an actual time since the amount of time an agent spends on each node, and the amount of time it takes an agent to migrate, is predictable.

The cost of an agent, a, visiting a node, m, following a certain path, p, is denoted $C_a(m,p)$. It is defined in terms of the weight of the agent, $W_a(m,p)$, and the amount of energy remaining on the node, E_m, as follows:

$$C_a(m,p) = \frac{W_a(m,p)}{E_m} \tag{1}$$

Note that as the amount of energy available on a node increases, the cost decreases and vice-versa. This is necessary to balance the residual energy of the nodes and thereby extend the lifetime of the network.

The weight of agent a, $W_a(m,p)$, is a function that returns the weight, in bytes, of agent a, when visiting node m, following path p. The weight of an agent depends on the size of its code and the amount of data that it carries. The size of its code is fixed, whereas the size of the data varies every hop and depends on what the application does upon visiting each node.

Finally, let P be the set of all paths and P_m be the set of paths that include node m, where $P_m \subseteq P$, we can formulate the following minimization problem:

$$\texttt{min} \left(\sum_{m \in \Omega} \sum_{p \in P_m} \frac{W_a(m,p)}{E_m} \right) \tag{2}$$

such that $\forall p \in P : |p| \leq t$ and the WSN is covered with density d.

3. A GENETIC ALGORITHM SOLUTION

Given the problem's complexity and the size of the search space($O(2^{2^{3t}})$), a heuristic is necessary. In addition, the sensor network's topology may change due to nodes running out of energy, desynchronizing, or being affected by external forces (e.g., being moved by the wind). Continuously monitoring the network topology is energy consuming. Thus, in order to solve the formulated minimization problem defined by equation 2, we propose a procedure containing two phases: an initial route planning phase done by a genetic algorithm and an in-network route adaptation phase that is done by each agent autonomously after it has been injected into the network.

Figure 1 contains the pseudo-code of the genetic algorithm (GA). It takes as input the definition of a sensor network Ω (i.e., the topology), the time constraint t (i.e., the maximum number of nodes of the longest route), the density d, and the specification of the varying size of the agents $W_a(m,p)$. The outcome is the number of agents and their routes, such that the collection is achieved in an energy-aware manner and within a time deadline. The GA can be divided into the following subparts: *Generation of initial population*, *Crossing*, *Mutation*, *Selection*, and *Adding of new individuals*.

```
01: GeneticAlgorithm(Ω,d,t,Wa(m,p))
02:    CreateInitialPopulation(N);
03:     For generation=1 to MaxGenerations
04:        Cross();
05:        Mutate();
06:        NaturalSelection();
07:        AddNewIndividuals(I);
08:     endFor
09:    return AgentRoutes
10: endGeneticAlgorithm
```

Figure 1: The Genetic Algorithm

Generation of initial population and the addition

of new individuals: An initial set, P, of solutions is generated. These solutions are a collection of random paths that cover the network, and are generated by repeatedly calculating paths until the whole network, Ω, has been covered with a density of d. Each path is selected by starting at the collection point and iteratively selecting a random neighbor until the collection point is reached again. Only those neighboring nodes whose minimum distances (in number of hops) to the collection point are not greater than the remaining time will be selected. Since the time constraint is expressed as the maximum number of nodes per path, comparing a nodes distance to the collection point with the deadline is trivial. All feasible neighbors are equally likely to be selected, and the initial size of P, $|P|$, is a design parameter.

Crossing and Mutation: At every iteration, a new generation is added to P by selecting x individuals to be crossed. Two individuals are crossed by taking a portion of each individual's math and combining them to form a new individual. Depending of how those individuals are selected, and which portion of their paths are crossed, new more efficient paths may be formed. Mutation is done by crossing an individual with itself.

Evaluation and Selection: All solutions are evaluated by the function being minimized in 2. Upon evaluation, $k_n\%$ of the worst individuals are randomly selected and removed from the population. Note that to avoid loosing the best solutions generated, the five best solutions are never removed, irrespective of k_n.

3.1 In-Network Route Adaptation

Since the topology of WSNs changes, mobile agents must adapt their routes during runtime. In particular, if an agent cannot successfully migrate from the i^{th} node of its route, to the $(i+1)^{th}$, it incrementally tries to migrate to the j^{th} node such that $j > (i+1)$. Given a dense WSN, it is likely that at least one of the nodes that follow the $(i+1)^{th}$ node on the agent itinerary will be in the i^{th} node's radio range, which allows the agent to resume its exploration. Furthermore, if the underlying mobile agent middleware supports multihop routing (e.g., geographic routing) such as Agilla [3], this solution is even more robust since an agent can migrate to any node in the WSN regardless of distance.

4. EVALUATION

To evaluate our system, we deployed it on an a wireless sensor network consisting of 25 Mica2 motes [1]running Agilla [3], a mobile agent middleware. Having a real implementation allowed us to define realistic $W_a(m,p)$ functions and empirically determine a translation from the deadline expressed in seconds to the maximum number of hops per path.

We measured the actual amount of time agent migration takes. We ran three different experiments with $t = 13$, 22, and 26. The GA generated twenty-five initial solutions, executed for five iterations, and added five new solutions every iteration thereafter. For each experiment, we averaged the three runs and plotted the percent of the network explored versus time. The results are shown in Figure 2.

As expected, Figure 2 shows that the genetic algorithm produces faster results with less variance when the maximum path length is more restrictive (shorter). With a time constraint of $t = 13$, 22, and 26, mobile agents spent an average of 14s, 22.33s, and 24s in the WSN, respectively. The

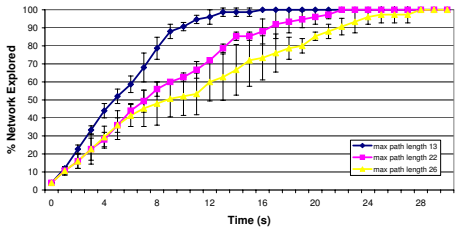

Figure 2: Percent Network Explored vs. Time

average time it takes to move an agent one hop is $1.233s$.

5. CONCLUSIONS

Software for WSNs needs to be more flexible. Mobile agents can provide this flexibility, but new algorithms are needed to control agent behavior. This paper addresses network exploration defined as visiting enough nodes in the network to achieve a certain level of coverage. We present a genetic algorithm that determines the number of agents necessary to explore the network and their paths. It includes contingency plans to account for the dynamic nature of WSNs. We show through experiments on a real WSN that our procedure is effective and that it is able to explore the network within the set deadline, while keeping energy consumption in check.

6. ACKNOWLEDGMENT

This research has been supported in part by the Generalitat de Catalunya, the Univ. of Girona, and the Univ. of California through a Girona fellowship, and by the US Office of Naval Research under MURI research contract N00014-02-1-0715.

7. REFERENCES

[1] Crossbow Technology, Inc. *MPR/MIB User's Manual*, 2004.

[2] C.-L. Fok, G.-C. Roman, and C. Lu. Mobile agent middleware for sensor networks: An application case study. In *Proc. of the 4th Int. Conf. on Information Processing in Sensor Networks (IPSN'05)*, pages 382–387. IEEE, April 2005.

[3] C.-L. Fok, G.-C. Roman, and C. Lu. Rapid development and flexible deployment of adaptive wireless sensor network applications. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'05)*, pages 653–662. IEEE, June 2005.

[4] G. Hackmann, C.-L. Fok, G.-C. Roman, C. Lu, C. Zuver, K. English, and J. Meier. Agile cargo tracking using mobile agents in wireless sensor networks. In *Proc. of the ACM SenSys*, page 303, 2005.

[5] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proc. of the 1st ACM Workshop on Wireless Sensor Networks and Applications*, pages 88–97, September 2002.

[6] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proc. of the ACM SenSys*, pages 13–24, 2004.