Optimal Itinerary Analysis for Mobile Agents in Ad Hoc Wireless Sensor Networks*

Hairong Qi
Electrical and Computer Engineering Department
University of Tennessee, Knoxville, TN 37996
hqi@utk.edu

Feiyi Wang
Advanced Networking Group
MCNC, Research Triangle Park, NC
fwang2@mcnc.org

Abstract

One of the most important problems studied in any sensor network is data fusion. Client/server paradigm has been a commonly used computing model in traditional distributed sensor networks (DSNs). However, the deployment of wireless sensor networks (WSNs) and its ad hoc nature have brought new challenges to the fusion task. For example, the advances in sensor technology allow better, cheaper, and smaller sensors to be used, which results in a much larger number of sensors deployed. On the other hand, sensors communicate through wireless networks where the network bandwidth is much lower than for wired communication. In this paper, we describe the usage of mobile agent for data fusion in WSNs. In this computing model, data stay at the local site, while the fusion process (code) is moved to the data sites. By transmitting the computation engine instead of data, network bandwidth requirement is largely reduced and the performance of the fusion process is more stable. One of the key problems discussed in this mobile-agent-based WSN (MAWSN) is how to plan the itinerary (or route) for a mobile agent in order to achieve progressive fusion accuracy. This paper presents a method to develop an optimal itinerary for mobile agent to fulfill the integration task while consuming minimum amount of resources, including time and power.

Keywords: mobile agent, ad hoc wireless sensor networks, data fusion, optimal itinerary

1 Introduction

Distributed sensor network (DSN) has become a very popular research topic due to its wide application spanning across civilian and military domain, including environmental monitoring (e.g. temperature sensing), generic object tracking (e.g. people or object locator), and surveillance in a large building or battlefield. The advances in sensor technology and ad hoc wireless networking have brought the study of DSN to a new stage — the emergence and spurs of wireless sensor networks (WSNs) [2, 7]. Recent development of mobile ad hoc networks (MANET) is just one example. It describes a distributed, mobile, wireless, multihop network that operates without the reliance on any existing infrastructure except the nodes themselves [4].

Even though it is economically feasible today to implement WSNs, there are several technical challenges that must be overcome before they can be used for the increasingly complex information gathering tasks. These tasks, such as battlefield surveillance, remote sensing, global awareness, etc., are usually time-critical, cover a large geographical area, and require reliable delivery of accurate information for their completion. The new challenges brought to the study of WSN include:

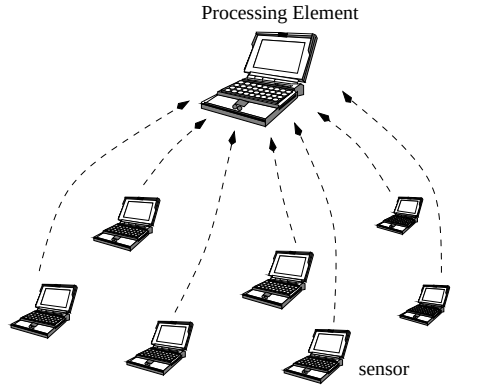
- data volumes being integrated are much larger due to the increasing amount of sensors being deployed;
- the communication bandwidth for wireless network is much lower;
- the environment is more unreliable, causing unreliable network connection and increasing the likelihood of input data to be in faulty; and
- fixed routing is impossible.

In traditional DSNs, data are collected by individual sensors, and then transmitted to a higher-level processing element which performs data fusion. During this process, large amount of data are moved around the network, as is the typical scenario in the client/server paradigm. In this paper, we adopt a new computing paradigm — mobile agents — we refer to this as mobile-agent-based WSN (MAWSN). In MAWSN, data stay at the local site, while the integration process (code) is moved to the data sites. By transmitting the computation engine instead of data, MAWSN offers the following important benefits:

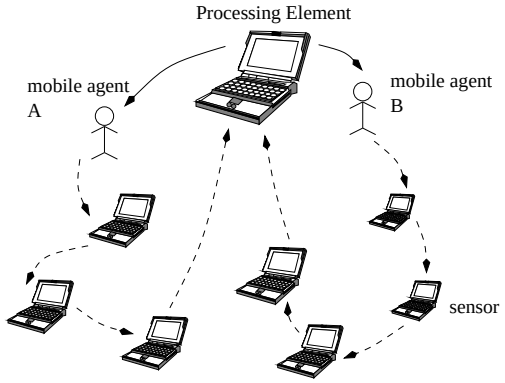
*This research was supported in part by DARPA under grant N66001-001-8946

- Network bandwidth requirement is reduced. Instead of passing large amount of raw data over the network through several round trips, only the agent with small size is sent. This is especially important for real-time applications and where the communication is through low-bandwidth wireless connections.
- Stability. Mobile agents can be sent when the network connection is alive and return results when the connection is re-established. Therefore, the performance of MAWSN is not much affected by the reliability of the network.

Figure 1 provides a comparison between DSN and MAWSN from architecture point of view.



(a) DSN



(b) MAWSN

Figure 1: Architecture comparison between DSN and MAWSN.

There are several issues related to the design of MAWSNs [14], such as the performance issue, the distributed data integration issue, etc. In this paper, we focus our discussion on how to design an optimal itinerary for mobile agent such that progressive integration accuracy can be achieved by consuming minimum amount of resources. These resources, including on-board sensor power, computing time, integration time, are critical in designing a high quality WSN.

2 Background

This section reviews the architecture of traditional DSN and the key characteristics of mobile agents.

A general DSN (Fig. 2) consists of a set of *sensor nodes*, a set of *Processing Elements* (PEs), and a *communication network* interconnecting the various PEs [8]. One or more sensors is associated with each PE. One sensor can report to more than one PE. A PE and its associated sensor(s) are referred to as a *cluster*. Data are transferred from sensors to their associated PE(s) where the data integration takes place. PEs can also coordinate with each other to achieve a better estimation of the environment and report to higher level PEs. In the context of this paper, we assume that the sensor field is a two-dimensional surface, and the sensor nodes are fixed once deployed.

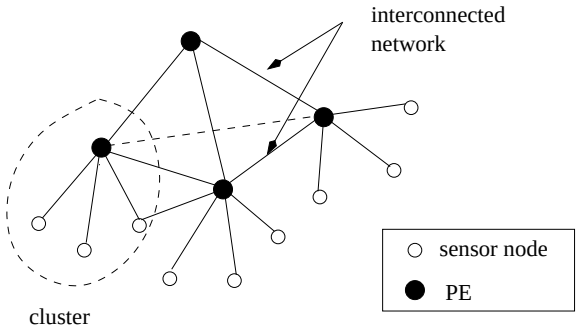


Figure 2: The architecture of a general DSN.

Generally speaking, mobile agent is a special kind of software which can execute autonomously. Once dispatched, it can migrate from node to node performing data processing autonomously, while software can typically only execute when being called upon by other routines.

Lange listed seven good reasons to use mobile agents [10], including reducing network load, overcoming network latency, robust and fault-tolerant performance, etc. Although the role of mobile agents in distributed computing is still being debated mainly because of the security concern [5, 11], several applications have shown clear evidence of benefiting from the use of mobile agents, such as E-commerce [3], distributed information retrieval and information dissemination [6, 9, 13, 15], etc.

In this paper, we use mobile agent in WSNs to perform data fusion.

3 Problem Formulation

We define the mobile agent as an entity of four attributes: identification, itinerary, data space, and method. These attributes are explained as follows:

- Identification: is in the format of 2-tuple (i, j), where i indicates the identification number of its dispatcher and j the serial number assigned by its dispatcher. Each mobile agent can be uniquely identified by this identification. We use $MA_{i,j}$ to indicate different mobile agents.
- Itinerary: includes itinerary information assigned by its associated PE when dispatched.
- Data space: agent's private data buffer which carries integration results and itinerary information.
- Method: the implementation of the data fusion algorithms.

Let PE_i represent a certain processing element with an identification i that is in charge of the surveillance of a certain area. Let $\{MA_{i,1}, \cdots, MA_{i,m}\}$ represent a group of m mobile agents dispatched by PE_i. Without loss of generality, we assume that each $MA_{i,j}$ visits the same number of sensor nodes, denoted by n. In this scenario, the benefit introduced by the use of mobile agents largely depends on the planning of the agent itinerary [1]. That is, we would like to choose an optimal itinerary that consumes the least amount of resources (time and power) in order to finish the fusion task.

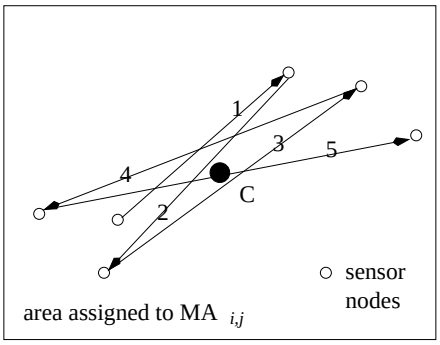
4 Itinerary Optimization

Itinerary can be determined either statically or dynamically. That is, it can be calculated either before the agent is dispatched or while the agent is migrating. Dynamic itinerary planning is more flexible, and can adapt to environmental changing (sensor ups and downs) in real time. However, since the itinerary is calculated on the fly, it also consumes more computation time and more power of the local sensor. On the other hand, although static itinerary cannot adapt to the network change, it is able to save both computation and power since the itinerary only needs to be calculated once. Computation-efficiency, power-efficiency, and flexibility are three parameters that cannot be satisfied at the same time. In this section, we first describe two ad hoc dynamic itinerary planning strategy. An optimal planning algorithm is proposed in the end.

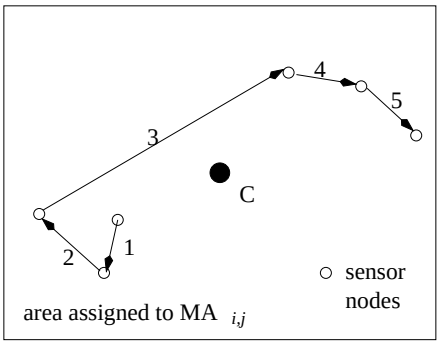
4.1 Ad Hoc Algorithms

The *itinerary* attribute in $MA_{i,j}$ is in the format of 3-tuple $(C_{i,j}, r_{i,j}, L_{i,j})$:

- $C_{i,j}$: the center coordinate of a sub-area to be traveled by $MA_{i,j}$;
- $r_{i,j}$: the radius of the sub-area;
- $L_{i,j}$: the list of destinations $MA_{i,j}$ needs to visit in one trip. Noted that $L_{i,j}$ is an unsorted list when $MA_{i,j}$ is dispatched the first time. A sorted $L_{i,j}$, however, can be reused after its first trip.



(a) GCF



(b) LCF

Figure 3: A comparison between GCF algorithm and LCF algorithm for itinerary planning under an extreme case.

Two ad hoc algorithms can be used to calculate the itinerary: local closest first (LCF) and global closest

first (GCF). Assume both algorithms start at the same sensor node closest to center $C_{i,j}$, LCF searches for the next node with the shortest distance to the current node, while GCF searches for the next closest node to center $C_{i,j}$. Under an extreme case that the n sensor nodes form two clusters centered at the two ends of the diameter in that sub-area, the itinerary planned by GCF can result in redundant fluctuation between these two clusters, while LCF handles this case effectively as shown in Fig. 3.

LCF is a two-part algorithm, described in detail as Algorithms 1 and 2:

Algorithm 1: LCF: itinerary planning algorithm for the first trip

```
Data  : C_{i,j}, r_{i,j}, L_{i,j}, S_k (current location, where
        k ∈ [0,n], k = 0 means MA_{i,j} is at PE_i)
Result: the next destination (D)
while true do
    switch the value of k do
        case k = 0
            find sensor node S with the smallest
            d(S, C_{i,j}) from L_{i,j};
            D = S;
        case k = n
            D = PE_i;
        otherwise
            find sensor node S with the smallest
            d(S_k, S) from the rest of L_{i,j};
            D = S;
    if D is active then
        migrate to D;
        k = k + 1;
        break;
    else
        delete D from L_{i,j};
        k = k + 1;
```

Algorithm 2: LCF: itinerary maintenance algorithm for following trips

```
Data  : C_{i,j}, r_{i,j}, L_{i,j}, S_k (current location, where
        k ∈ [0,n], k = 0 means MA_{i,j} is at PE_i)
Result: the next destination (D)
while true do
    switch the value of k do
        case k = n
            D = PE_i;
        otherwise
            D = S_{k+1};
    if D is active then
        migrate to D;
        k = k + 1;
        break;
    else
        delete D from L_{i,j};
        k = k + 1;
```

4.2 Optimal Itinerary

The problem of optimally planning the itinerary for mobile agents has been discussed in [12] from probability point of view, where n sites are given at which a certain task might be successfully performed. The probability of success at each site is p_i. In this paper, we study the itinerary problem within the context of data fusion where computation time and power consumption are the two major concerns. On the other hand, as large amount of sensors are deployed, redundancy in the sensor readouts are used to provide error tolerance. Multiresolution techniques are popularly used such that when the accuracy requirement is not high, processing can be carried out at a coarser resolution to save both the computation time and the power consumed. Furthermore, when mobile agent migrating around the sensor network and accumulating sensor readouts, if the accuracy of the result has reached the requirement of a certain task, the agent can return to the processing center directly without finishing the entire trip. Due to the complexity of this approach, it is better applied before the agent is dispatched. In another word, it is suggested that the optimal itinerary be used to derive static itinerary.

Let $\delta_{i,j}$ be the accuracy requirement for a specific task (e.g. accuracy of range of detected target) that agent $MA_{i,j}$ carries out. Each agent plans to visit $k = 1 \cdots n$ sensors. We assume each sensor measures the same set of parameter(s), and the readout of each parameter is a range of real numbers $[a_k, b_k]$. Let p_k be the percentage of sensor readout that includes true values of the environment, q_k be the percentage of power remaining on the sensor node. Let t be the processing time spent at each sensor node and s be the power consumed at each sensor which are the same for all sensors. Let us also assume that the processing time is much longer than the agent travelling time, so that it can be ignored. We still use $L_{i,j}$ to represent the list of sensor nodes that $MA_{i,j}$ should visit. As the migration goes on, the width of parameter estimation $[a_k, b_k]$ should be getting narrower and narrower, while the accuracy approaching $\delta_{i,j}$. We use the change made between adjacent readout range $|(b_k - a_k) - (b_{k-1} - a_{k-1}|/|b_{k-1} - a_{k-1}|$ to represent the accuracy. If the accuracy has reached $\delta_{i,j}$, then the mobile agent does not need to go through the rest of the sensor nodes, instead, it can return to the PE directly which saves both migration time and network bandwidth.

An optimal list of nodes $L_{i,j}$ is searched such that the cost of computation time and the relative power consumption with respect to each node itself reaches the minimum. An objective function as Eq. 1 is derived, where H_T is the time consumed and H_P is the relative power consumed, α is a positive real number that is less than 1. It indicates the tradeoff between H_T and H_P.

$$H(L_{i,j}) = \alpha H_T + (1-\alpha)H_P \quad (1)$$

where

$$H_T = -\sum_{k=1}^{n} tp_k$$

and

$$H_P = -\sum_{k=1}^{n} sq_k$$

This optimization problem can be easily solved by genetic algorithm. With the optimal itinerary $L_{i,j}$ obtained, Algorithm 3 describes how it is used subject to the requirement of accuracy.

Algorithm 3: Optimal itinerary planning subject to accuracy requirement

```
Data : L_{i,j} (optimal itinerary), δ_{i,j} (accuracy requirement), [a_k, b_k] (sensor readout of a specific parameter), where k = 1 ··· n, w_old (the width of the previous sensor readout)
w_old = ∞;
k = 1;
while k <= n do
    while L_{i,j}[k] is not active do
        k = k + 1;
    end
    migrate to L_{i,j}[k];
    obtain the sensor readout as [a_k, b_k];
    if |(b_k − a_k) − w_old| / w_old < δ_{i,j} then
        break;
    else
        w_old = b_k − a_k;
        k = k + 1;
    end
end
return to the processing center;
```

5 Summary

This paper describes the use of the mobile agent paradigm for data fusion in WSNs. We focus our discussion on optimal itinerary design of the mobile agent. Two ad hoc algorithms are proposed to compare with the performance of the optimal itinerary.

References

[1] P. Alimonti, F. Lucidi, and S. Trigila. *Agent Technology for Communication Infrastructures*, chapter 17. How to Move Mobile Agents, pages 222–233. John Wiley, Chichester, 2001.

[2] John Byers and Gabriel Nasser. Utility-based decision-making in wireless sensor networks. Technical Report 2000-014, Computer Science Dept., Boston University, January 2000.

[3] P. Dasgupta, N. Narasimhan, L. E. Moser, and P. M. Melliar-Smith. Magnet: mobile agents for networked electronic trading. *IEEE Transactions on Knowledge and Data Engineering*, 11(4):509–525, July/August 1999.

[4] James A. Freebersyser and Barry Leiner. *Ad Hoc Networking*, chapter 2: A DoD Perspective on Mobile Ad Hoc Networks, pages 29–51. Addison Wesley, 2000.

[5] C. G. Harrison, D. M. Chess, and A. Kershenbaum. Mobile agents: are they a good idea? Technical Report RC 19887, IBM Thomas J. Watson Research Center, March 1995. http://www.research.ibm.com/massive/mobag.ps.

[6] M. Hattori, N. Kase, A. Ohsuga, and S. Honiden. Agentb-based driver's information assistance system. *New Generation Computing*, 17(4):359–367, 1999.

[7] Wendi R. Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Fifth ACM/IEEE MobiCom*, pages 1–12, Seattle, WA, August 1999. ACM/IEEE.

[8] S. S. Iyengar, D. N. Jayasimha, and D. Nadig. A versatile architecture for the distributed sensor integration problem. *IEEE Transactions on Computers*, 43(2):175–185, February 1994.

[9] J. Kay, J. Etzl, G. Rao, and J. Thies. Atl postmaster: a system for agent collaboration and information dissemination. In *Proceedings of the 2nd International Conference on Autonomous Agents*, pages 338–342, Minneapolis, MN, 1998. ACM.

[10] D. B. Lange and M. Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, March 1999.

[11] D. Milojicic. Mobile agent applications. *IEEE Concurrency*, pages 80–90, July-September 1999.

[12] K. Moizumi and G. Cybenko. The travelling agent problem. Technical report, Dartmouth College, Hanover, NH, February 1998.

[13] T. Oates, M. V. N. Prasad, and V. R. Lesser. Cooperative information-gathering: a distributed problem-solving approach. *IEE Proceedings - Software Engineering*, 144(1):72–88, February 1997.

[14] H. Qi, S. S. Iyengar, and K. Chakrabarty. Distributed multi-resolution data integration using mobile agents. In *IEEE Aerospace Conference*, Big Sky, MT, March 2001. IEEE.

[15] J. S. Wong and A. R. Mikler. Intelligent mobile agents in large distributed autonomous cooperative systems. *Journal of Systems and Software*, 47(2):75–87, 1999.