Distributed Multi-Resolution Data Integration Using Mobile Agents *†

Hairong Qi
ECE Department
The University of Tennessee
319 Ferris Hall
Knoxville, TN 37996-2100
865-974-8527
hqi@utk.edu

S. Sitharama Iyengar
CS Department
Louisiana State University
298 Coates Hall
Raton Rouge, LA 70803
225-388-1495
iyengar@bit.csc.lsu.edu

Krishnendu Chakrabarty
ECE Department
Duke University
130 Hudson Hall, Box 90291
Durham, NC 27708
919-660-5244
krish@ee.duke.edu

Abstract — We describe the use of the mobile agent paradigm to design an improved infrastructure for data integration in Distributed Sensor Network (DSN). We use the acronym MADSN to denote the proposed Mobile-Agent-based DSN. Instead of moving data to processing elements for data integration, as is typical of a client/server paradigm, MADSN moves the processing code to the data locations. This saves network bandwidth and provides an effective means for overcoming network latency, since large data transfers are avoided. We study two important problems related to MADSN design — the distributed integration problem, and the optimum performance problem. Compared to DSNs, a mobile-agent implementation of multi-resolution data integration saves up to 90% of the data transfer time. For a given set of network parameters, we analyze the conditions under which MADSN performs better than DSN and determine the condition under which MADSN reaches its optimum performance level.

TABLE OF CONTENTS

1 INTRODUCTION

Distributed Sensor Networks (DSNs) have recently emerged as an important research area [6, 7, 9, 14, 18]. This development has been spurred by advances in sensor technology and computer networking. Even though it is economically feasible today to implement DSNs, there are several technical challenges that must be overcome before DSNs can be used for today's 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.

Wesson et al [18] were among the first to propose the design of DSNs. Since then, several efficient DSN architectures have been presented in the literature, including the hierarchical and committee organization [18], the flat tree network [7, 14], the deBruijn based network [6], and the multi-agent fusion network [9]. While improving the performance of DSNs in different aspects, all these approaches use a common network computing model: the client/server model, which supports many distributed systems, such as remote procedure calling (RPC) [2], common object request broker architecture (CORBA) [1, 17], etc. In the client/server model, the client (individual sensor) sends data to the server (processing element) where data processing tasks are carried out.

Recent advances in sensor technology allow better, cheaper, and smaller sensors to be used in both military and civilian applications, especially when the environment is harsh, unreliable, or even adversarial. A large number of sensors are usually deployed in order to achieve quality through quantity. On the other hand, sensors typically communicate through wireless networks where the network bandwidth is much lower than for wired communication. These issues bring new challenges to the design of DSNs: First, data volumes being integrated are much larger; Second, the communication bandwidth for wireless network is much lower; Third, the environment is more unreliable, causing unreliable network connection and increasing the likelihood of input data to be in faulty.

In this paper, we design an improved DSN architecture using mobile agents — we refer to this as mobile-agent-based DSN (MADSN). In traditional DSNs, data are collected by individual sensors, and then transmitted to a higher-level processing

* 0-7803-6599-2/01/$10.00©2001 IEEE
† This research was supported in part by DARPA under grant N66001-001-8946

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2001		**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Distributed Multi-Resolution Data Integration Using Mobile Agents	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Defense Advanced Research projects Agency,3701 North Fairfax Drive,Arlington,VA,22203-1714**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**9**	

element which performs sensor fusion. During this process, large amounts of data are moved around the network, as is the typical scenario in the client/server paradigm. MADSN adopts a new computation paradigm: 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, MADSN offers the following important benefits:

- Network bandwidth requirement is reduced. Instead of passing large amounts 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.
- Better network scalability. The performance of the network is not affected when the number of sensor is increased. Agent architectures that support adaptive network load balancing could do much of a redesign automatically [16].
- Extensibility. Mobile agents can be programmed to carry task-adaptive fusion processes which extends the capability of the system.
- 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 MADSN is not much affected by the reliability of the network.

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

The organization of this paper is as follows: Section discusses the definition of mobile agents and application examples that benefit from using mobile agents. It also defines the two problems studied in the design of MADSN. Section first reviews the multi-resolution data integration algorithm implemented under traditional DSN, then describes its implementation using mobile agents. A case study is provided. Section 4.2 compares the performance of DSN and MADSN. For a given set of parameters, it derives the condition under which MADSN performs better than DSN, also the condition under which MADSN reaches its optimum performance level. Section 4.2 summarizes the paper and draws conclusions.

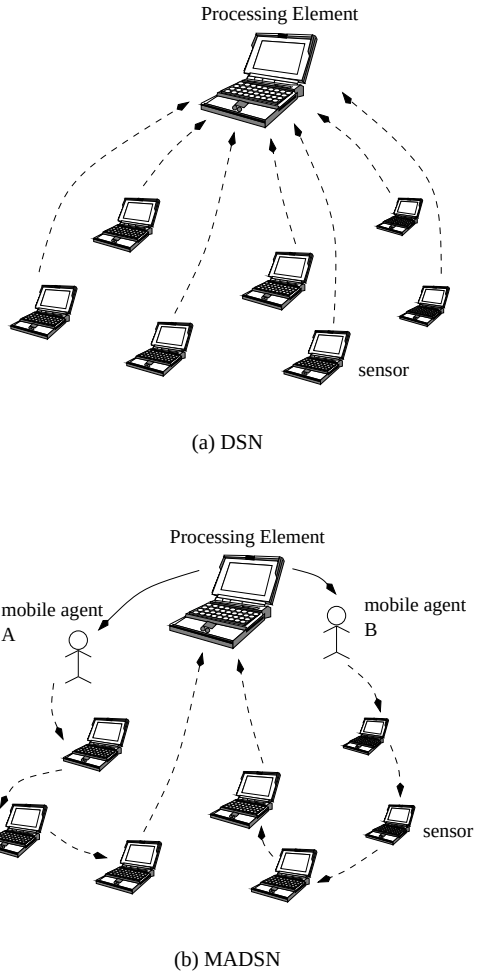

Figure 1: Architecture comparison between DSN and MADSN.

2 BACKGROUND

This section reviews the basic DSN architecture and the key characteristics of mobile agents. The problems studied in this paper are formally defined at the end of the section.

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 [6]. 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.

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 [11], 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 [4, 12], several applications have shown clear evidence of benefiting

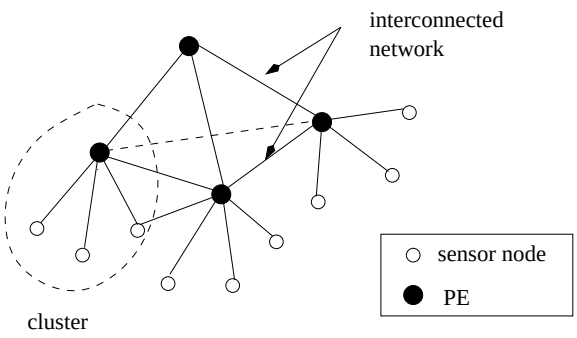

Figure 2: The architecture of a general DSN.

from the use of mobile agents, such as E-commerce [3], distributed information retrieval and information dissemination [5, 8, 13, 19], etc.

In this paper, we use mobile agent in DSNs to perform multi-resolution data integration and fusion. Problems to be studied are defined in the following section.

3 PROBLEM STATEMENT

We define the mobile agent as an entity of five attributes: identification, itinerary, data, method, and interface. 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: agent's private data buffer which carries integration results and itinerary information.
- Method: the implementation of algorithms. In MADSN, the key method is the multi-resolution data integration algorithm for sensor fusion.
- Interface: provides interface functions for agent and processing element to communicate with each other.

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. The parameters m and n are related in the sense that their product mn equals the number of sensor nodes in the field. The problems studied in this paper are formally defined as follows:

Data integration problem: At each sensor site, what kind of data processing should be conducted and what integration results should be carried with the mobile agent?

Optimum performance problem: How to balance the value of m and n, such that the performance of MADSN is superior to DSN.

4 MULTI-RESOLUTION INTEGRATION ALGORITHM

As mentioned in Sec. , MADSN must respond to the challenges of a large amount of sensor nodes and higher probability of faulty sensors. More sensor nodes can increase the computation load, while more faulty sensors can cause the integration results to be unreliable. Algorithms are therefore sought which should not be significantly affected by network scaling, and yet provide better performance and higher fault tolerance. This section first reviews the original Multi-Resolution Integration (MRI) algorithm proposed for DSNs [15]. Enhancements to the basic MRI algorithm are then described in order to take advantage of mobile agents to achieve better network scalability and fault tolerance. The enhancements involve a multi-resolution analysis of individual sensor readout to generate a simple function (the overlap function) at the sensor site, followed by an integration of the simple functions at the processing element. Compared to the MRI algorithm in traditional DSNs, where the integration of individual sensor readout (carried out at the processing element) is followed by the multi-resolution analysis of the integrated simple function, the mobile agent implementation of MRI algorithm reduces the data transfer time by as much as 90%.

4.1 *Original MRI Algorithm in DSNs*

The original MRI algorithm was proposed by Prasad, Iyengar and Rao in 1994 [15]. The idea essentially consists of constructing a simple function (the overlap function) from the outputs of the sensors in a cluster and resolving this function at various successively finer scales of resolution to isolate the region over which the correct sensors lie. Each sensor in a cluster measures the same parameters. It is possible that some of them are faulty. Hence it is desirable to make use of this redundancy of the readings in the cluster to obtain a correct estimate of the parameters being observed. We first review several relevant definitions.

An **abstract sensor** is defined as a sensor that reads a physical parameter and gives out an abstract interval estimate which is a bounded and connected subset of the real line. We classify abstract sensors into two categories: **correct sensors** and **faulty sensors**. A **correct sensor** is an abstract sensor whose interval estimate contains the actual value of the parameter being measured. Otherwise, it is a **faulty sensor**. A faulty sensor is **tamely faulty** if it overlaps with a correct sensor, and is **wildly faulty** if it does not overlap with any correct sensor.

$$\chi_j(x) = \begin{cases} 1, & \text{if } x \text{ is in } I_j, \text{ and } 1 \le j \le n \\ 0, & \text{if } x \text{ is not in } I_j, \text{ and } 1 \le j \le n \end{cases} \quad (1)$$

Let sensors $S_1, \cdots, S_n$ feed into a processor P. Let the abstract interval estimate of S_j be I_j $(1 \le j \le n)$, the closed interval $[a_j, b_j]$ with end points a_j and b_j. The **characteristic function** χ of the jth sensor S_j is defined in Eq. (1).

Let $\Omega(x) = \sum_{j=1}^{n} \chi_j(x)$ be the **overlap function** of the n abstract sensors. For each $x \in R$, $\Omega(x)$ gives the number of sensor intervals in which x lies; that is, the number of intervals overlapping at the x. **Crest** is a region in the overlap function with the highest peak and the widest spread. Figure 3 illustrates the overlap function for a set of 7 sensors. The notion of the overlap function allows us to make the following key observations:

- Tamely faulty sensors cluster around correct sensors and create high and wide (maximal) peaks in the profile of $\Omega(x)$.
- Wildly faulty sensors on the other hand do not overlap with correct sensors, and therefore contribute to smaller and narrower peaks.

Therefore, the actual value of the parameter being measured lies within regions over which the maximal peaks of $\Omega(x)$ occur.

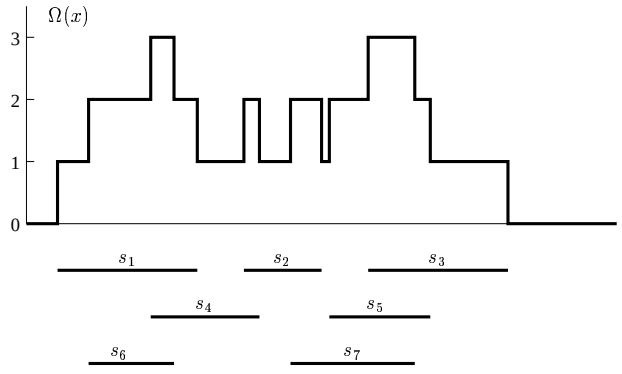

Figure 3: The overlap function for a set of 7 sensors.

Multi-Resolution Analysis of the Overlap Function — Multi-resolution analysis provides a hierarchical framework for interpreting the overlap function. It is natural and more efficient to first analyze details at a coarse resolution and then increase the resolution for only the region of interest.

Given a sequence of increasing resolutions $(2^{-c}, 2^{-c+1}, \cdots, 2^0)$, where c is a positive integer, we define the difference of function $f(x)$ at resolution 2^{-c+1} and resolution 2^{-c} as the details of $f(x)$ at resolution 2^{-c+1}. The algorithm is described in Algorithm 1.

This procedure results in the isolation of those regions of the real line over which the overlap function $\Omega(x)$ has a maximum value, corresponding to high degree of overlapping of individual sensor readouts. The algorithm is optimal, since the overall time required is $O(n \log n)$, which is the time required to maintain $\Omega(x)$. This algorithm is also robust, satisfies a Lipschitz condition [10], which ensures that minor changes in the input intervals cause only minor changes in the integrated result. Figure 4 illustrates the multi-resolution analysis procedure.

Algorithm 1: Multi-resolution analysis of the overlap function.

Data : $\Omega(x)$, 2^k, $(-c \le k \le 0)$, assuming the coarsest resolution is 2^{-c}, the highest resolution is 2^0; the initial integration interval $[A, B]$

Result : the final crest $[\gamma_l, \gamma_h]$ under resolution 2^k, where γ_l and γ_h are the lower and higher bounds of the crest respectively

$t = -c$;
while $t <= k$ **do**
  resolve $\Omega(x)$ at resolution 2^t by sampling it over the interval $[A, B]$ at points $n2^{-t}$, $(A/2^{-t} \le n \le B/2^{-t})$, to obtain $\Omega_t(x)$;
  select the highest peaks from $\Omega_t(x)$;
  choose from these peaks the one with the widest spread $[A_t, B_t]$, which is a crest;
  $\Omega(x) = \Omega_t([A_t, B_t])$;
  $A = A_t, B = B_t$;
  $t = t + 1$;
end
$\gamma_l = A, \gamma_h = B$

4.2 *MRI Implementation Using Mobile Agents*

In a distributed sensor network (DSN), all readouts from the sensor nodes are sent to their corresponding PEs, where the overlap function at the *finest* resolution is first generated, and the multi-resolution analysis procedure is then applied to find the crest at the *desired* resolution.

In a Mobile-Agent-based DSN (MADSN), the mobile agents migrate among the sensor nodes and collect readouts. Therefore, $MA_{i,j}$ always carries a *partially* integrated overlap function which is accumulated into a final version at PE_i after all the mobile agents return. During this process, if MADSN applies the multi-resolution analysis method in the same way as DSN does, that is, letting $MA_{i,j}$ carry the partially integrated overlap function in its finest resolution and then use multi-resolution analysis (MRA) to find the crest at desired resolution at PE_i, the advantages of mobile agents will be nullified because of heavy data migration.

We enhance the basic multi-resolution integration (MRI) algorithm for MADSNs and present a more efficient implementation. The key concept underlying the enhanced algorithm is

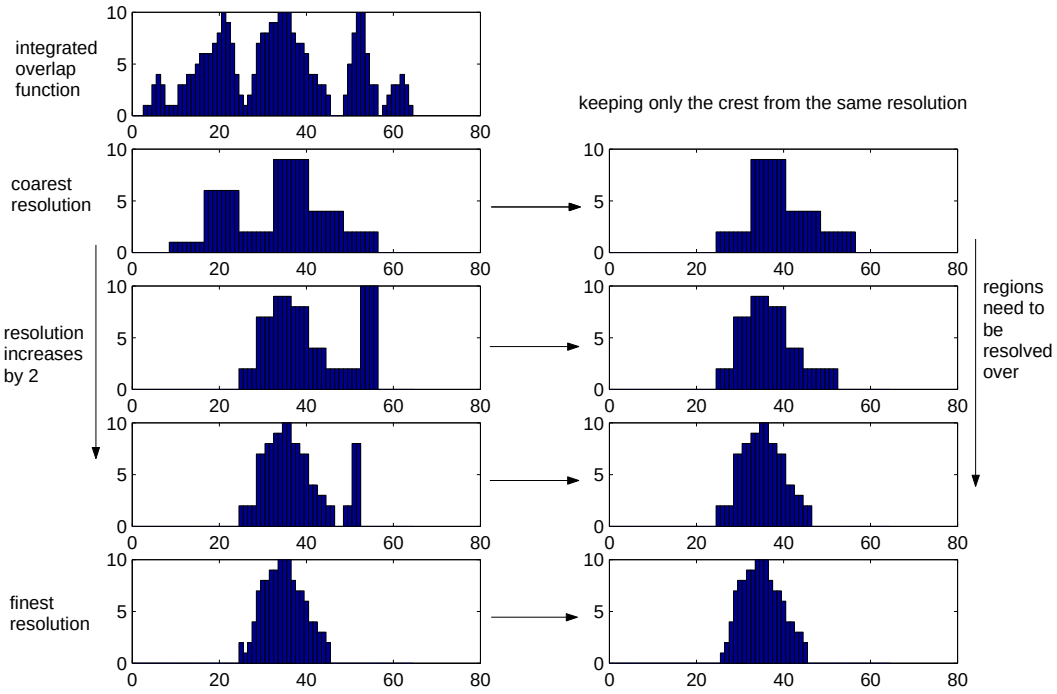

Figure 4: An overlap function $\Omega(x)$ and its appearance at different resolutions (The shaded region indicates the region needs to be resolved over).

that MRI is applied *before* accumulating the overlap function. A 1-D array, $\omega_{i,j}$, can serve as an appropriate data structure to represent the partially-integrated overlap function carried by $MA_{i,j}$. If the size of $\omega_{i,j}$ is s_{2^0} with 2^0 resolution, then with resolution 2^k (k times coarser than 2^0), the size of $\omega_{i,j}$ is s_{2^0}/k, that is, k times less than s_{2^0}. Algorithms 2-4 describe the procedure in detail. A case study is provided as well for better illustration.

Algorithm 2: Modified MRI algorithm for MADSN - before $MA_{i,j}$ leaves PE_i

Data : integration interval $[A_i, B_i]$, highest resolution 2^0, desired resolution 2^k

Result : array $\omega_{i,j}$ to hold partially-integrated overlap function

$s = \frac{B_i - A_i + 1}{2^{-k}}$;

initialize $\omega_{i,j}$ as a zero vector with s elements;

Algorithm 3: Modified MRI algorithm for MADSN - $MA_{i,j}$ at sensor node

Data : $\omega_{i,j}$, 2^k, readout from the abstract sensor $[a, b]$ (a bounded connected set of real numbers)

Result : $\omega_{i,j}$

find the smallest multiple of 2^{-k}, d_{min}, such that $d_{min} \geq a$;

find the largest multiple of 2^{-k}, d_{max}, such that $d_{max} \leq b$;

increase elements $\omega_{i,j}[\frac{d_{min}}{2^{-k}} : \frac{d_{max}}{2^{-k}}]$ by 1;

Algorithm 4: Modified MRI algorithm for MADSN - $MA_{i,j}$ back to PE_i

Data : $\omega_{i,j}$, 2^k, m is the total number of agents

Result : the final crest $[\gamma_l, \gamma_h]$ under resolution 2^k

create ψ_i as a zero vector of size $(B_i - A_i + 1)$;

$j = 2$;

while $j <= m$ **do**

  accumulate $\omega_{i,j}$ to $\omega_{i,1}$;

  $j = j + 1$;

end

$j = 0$;

while $j < \frac{B_i - A_i + 1}{2^{-k}}$ **do**

  $\psi_i[j : j + 2^{-k} - 1] = \omega_{i,1}[j/2^{-k}]$;

  $j = j + 2^{-k}$;

end

select the highest peak of ψ_i. If there are multiple peaks with the same height, then all the peaks should be selected;

choose from these peaks the one with the widest spread $[\gamma_l, \gamma_h]$, which is a crest;

Case Study — We present a case study to illustrate the MADSN-based MRI algorithm. Suppose PE_i has 10 sensor nodes $(S_1, \cdots, S_{10})$, migrated by 2 mobile agents with $MA_{i,1}$ covering S_1 to S_5, and $MA_{i,2}$ covering S_6 to S_{10}. The readouts of sensors at time t are listed in Fig. 5. The integration interval $[A_i, B_i]$ is $[1, 64]$. The overlap function at its highest resolution then has 64 elements.

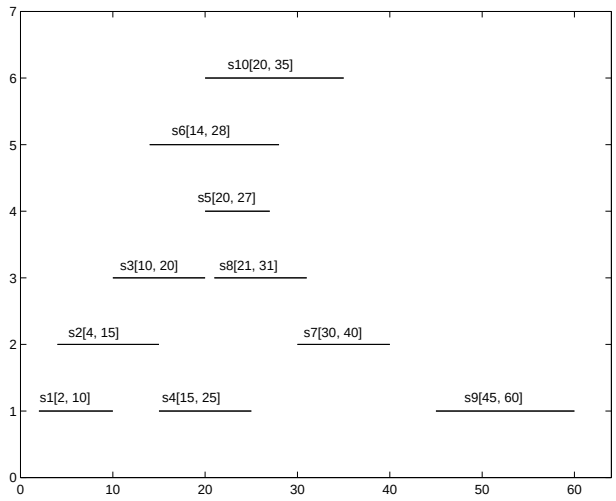

Figure 5: Readouts from 10 sensor nodes at time t.

	$\frac{d_{min}}{2^{-k}}$	$\frac{d_{max}}{2^{-k}}$	partially integrated $\omega_{i,1}$
S_1	1	1	$[0,1,0,0,0,0,0,0]$
S_2	1	1	$[0,2,0,0,0,0,0,0]$
S_3	2	2	$[0,2,1,0,0,0,0,0]$
S_4	2	3	$[0,2,2,1,0,0,0,0]$
S_5	3	3	$[0,2,2,2,0,0,0,0]$

Table 1: Tracing the change of $\omega_{i,1}$ generated by $MA_{i,1}$.

If the desired resolution is 2^{-3} (or eight times coarser than the finest resolution), according to Algorithm 2, an array $\omega_{i,j}$ with $8 = 64/8$ elements will be created and initialized by each mobile agent. Tables 1 and 2 list the step-by-step execution for each agent according to Algorithm 3.

According to Algorithm 4, the final integrated ψ_i will be $[0,2,3,5,2,1,1,1]$. Compared to the results from DSN, as shown in Fig. 6, they are exactly the same. If we define the unit data transfer time as the time spent for one $MA_{i,j}$ migrating from one node to another, carrying a one-element array, then MADSN spends $8 \times (5+2) = 56$ units of time (assuming $MA_{i,1}$ and $MA_{i,2}$ are executed in parallel when migrating from node to node or from PE_i to node, 8×5, and in serial when returning to PE_i, 8×2), while DSN spends $64 \times 10 = 640$ units of time. Hence, MADSN offers a save of up to 91.25% of data transfer time in this case.

	$\frac{d_{min}}{2^{-k}}$	$\frac{d_{max}}{2^{-k}}$	partially integrated $\omega_{i,2}$
S_6	2	3	$[0,0,1,1,0,0,0,0]$
S_7	4	5	$[0,0,1,1,1,1,0,0]$
S_8	3	3	$[0,0,1,2,1,1,0,0]$
S_9	6	7	$[0,0,1,2,1,1,1,1]$
S_{10}	3	4	$[0,0,1,3,2,1,1,1]$

Table 2: Tracing the change of $\omega_{i,2}$ generated by $MA_{i,2}$.

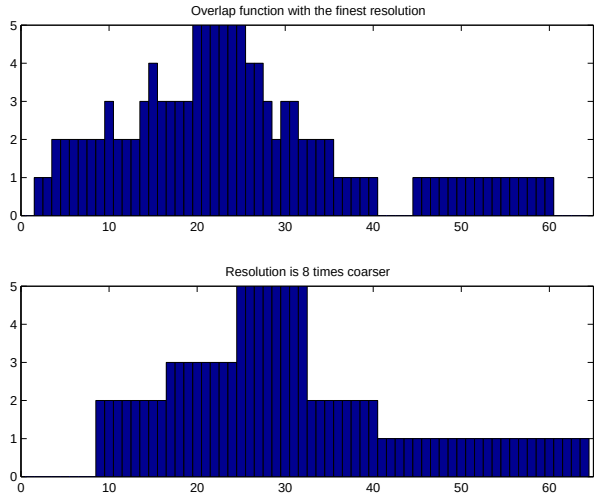

Figure 6: The overlap function at its highest resolution and the version with 8 times coarser resolution.

5 Performance Comparison

The case study shows that while obtaining the same integration results, MADSN saves 91.25% of data transfer time compared to DSN. However, this does not necessarily mean that MADSN is always better than DSN since MADSN also introduces overhead, such as the agent creation and dispatch time. On the other hand, DSN needs to transfer data files to PE_i which also causes overhead due to file accesses. In this section, we analyze the relative performances of DSN and MADSN, and determine conditions under which an MADSN is more efficient than a DSN. These conditions are determined by the network transfer rate v_n, the data processing rate v_d, the data file size s_f, the mobile agent data size s_a (including overlap function array size and the itinerary list size), overhead of agent o_a, overhead of file access o_f, the number of sensor nodes p, and thus the balance between the number of agents m and the number of sensor nodes each agent migrates n (Notice that $p = m \times n$). Equations (2) and (3) are two formulas estimating the execution time for MADSN (t_{madsn}) and DSN (t_{dsn}). In both equations, the three components calculate the data transfer time, the overhead, and the data processing/integration time respectively.

$$t_{madsn} = \frac{(m+n)s_a}{v_n} + mo_a + \frac{(m+n-1)s_a}{v_d} \quad (2)$$

$$t_{dsn} = \frac{mns_f}{v_n} + mno_f + \frac{(mn-1)s_f}{v_d} \quad (3)$$

We use m as the variable. Assume k and j are positive scalars, and $s_f = ks_a$, $o_f = jo_a$, $v'_n = 1/v_n$, $v'_d = 1/v_d$, in order to ensure that $t_{madsn} \leq t_{dsn}$, Eq. (4) must be satisfied, that is, m must be chosen within a range.

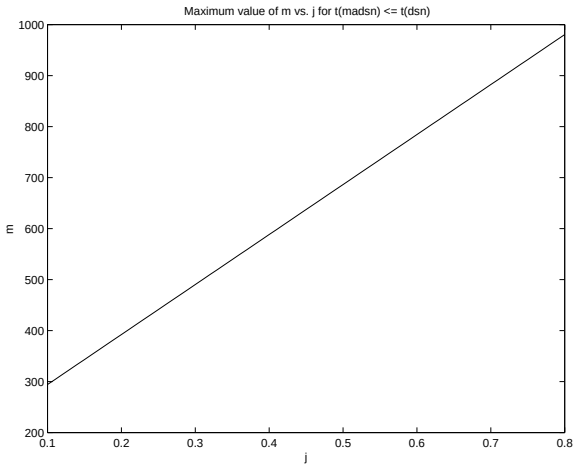

Figure 7: Performance evaluation between DSN and MADSN: m vs. j.

$$(s_a v'_n + o_a + s_a v'_d)m^2 - (s_a v'_d - k s_a v'_d + k p s_a v'_n + k p s_a v'_d + j p o_a)m + p s_a v'_n + p s_a v'_d \leq 0 \quad (4)$$

Table 3 listed some typical parameter values and the corresponding performance. We evaluate the performance variation of MADSN with respect to relationships between m and j, m and v'_n, and m and p. m is the number of nodes migrated by each mobile agent. j is the overhead ratio between DSN and MADSN. v'_n is the reciprocal of network transfer rate. p is the total number of sensor nodes. These parameters play a more important role than others. Figures 7-9 demonstrate the performance variation with respect to m and j.

Figure 7 shows a profile of the maximum value of m satisfying Eq. 4 when changing the overhead ratio between MADSN and DSN, j. Suppose the size of agent is 1KB, the overhead of agent is 0.5s (including agent creation time), the network transfer rate is 100Kbps, data processing rate is 100Mbps, the number of sensor nodes is 1000, and the data size is 10KB.

If we fix j at 0.25, that is, the overhead of file access is one fourth of the overhead of mobile agent, the corresponding maximum m satisfying Eq. 4 is then 441 according to Fig. 7. By changing m from 1 to 441, we generate the performance curves for MADSN and DSN using the execution time: t_{madsn} and t_{dsn} as shown in Figs. 8 and 9.

Figure 8 shows the variation of t_{dsn} with respect to the number of mobile agents m. It is a straight line since t_{dsn} is independent of the number of mobile agents and the total number of sensor nodes is a constant. Figure 9 illustrates the variation of t_{madsn} with respect to m. The execution time t_{madsn} reaches its minimum when m is 4. Note that even though in the range of $m \in [1, 441]$, t_{madsn} is always less than t_{dsn}, after a decreasing segment at the very beginning, and reaching a minimum when $m = 4$, t_{madsn} starts to increase. This is because of the overhead from mobile agent: the more agents used, the heavier the overhead, the longer execution time needed; on the other hand, the less the agents, the lighter the overhead, but the longer the migration time.

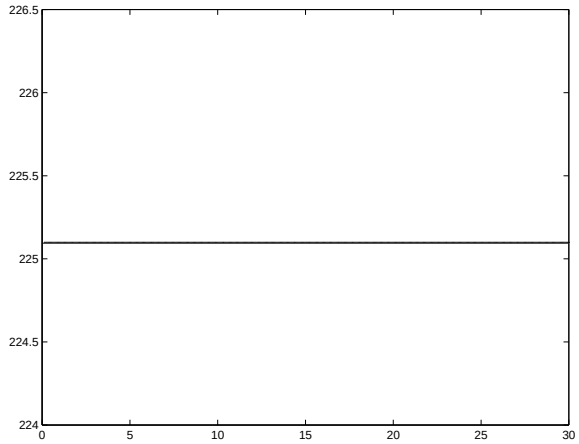

Figure 8: Execution time for DSN (t_{dsn}) with respect to m with $p = 1000$, v_t = 100Kbps, and $j = 0.25$.

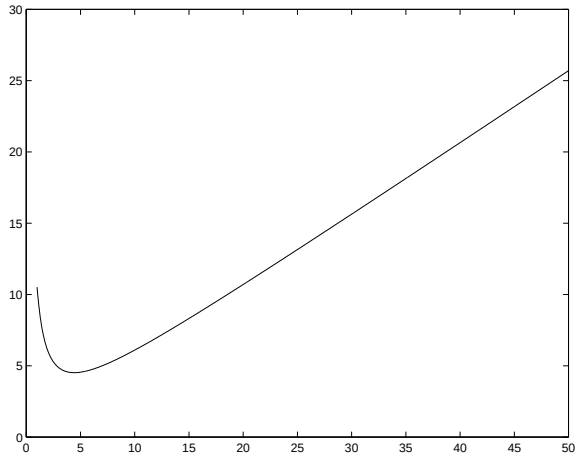

Figure 9: Execution time for MADSN (t_{madsn}) with respect to m with $p = 1000$, v_t = 100Kbps, and $j = 0.25$.

6 CONCLUSIONS

This paper describes the use of the mobile agent paradigm to design an improved infrastructure for sensor fusion in distributed sensor network (DSN). We use the acronym MADSN to denote the proposed mobile-agent-based DSN. Compared to the traditional client/server paradigm, where data are moved from the client to the processing center, MADSN moves the processing code to the data locations. This saves

Parameters	case 1	case 2	case 3
size of agent (s_a)	1K	1K	1K
ratio $k = \frac{s_f}{s_a}$	10	10	10
data processing rate (v_d)	100Mbps	100Mbps	100Mbps
overhead of agent (o_a)	0.5s	0.5s	0.5s
ratio $j = \frac{o_f}{o_a}$	0.25	0.25	0.25
network transfer rate (v_n)	100Kbps	500 Kbps	500Kbps
total number of sensor nodes (p)	1000	1000	3000
optimal number of agents (m)	4	3	4
execution time in DSN (t_{dsn})	225.1s	145.2s	435.3s
execution time in MADSN (t_{madsn})	4.5s	2s	3.5s
execution time saved ($\frac{t_{dsn}-t_{madsn}}{t_{dsn}} \times 100\%$)	98%	98.6%	99.2%

Table 3: Summary of performance comparison between DSN and MADSN.

network bandwidth and provides and effective means for overcoming network latency, since large data transfers are avoided. We studied two important problems related to MADSN design: the distributed integration problem, and the optimum performance problem. We show that MADSN is not always better than DSN, since the involvement of mobile agents adds overhead. We analyze the conditions under which MADSN performs better than DSN and the conditions under which MADSN achieves its optimum performance. From the performance comparison results in Table 3, we can see that for all three cases, MADSN saves up to 98% of execution time which is mainly contributed from the time saved for transferring raw data. We conclude that mobile agent paradigm is an effective approach for distributed computing, especially when large amount of data transfer is involved, which is the case in distributed sensor networks.

References

[1] S. Baker. Corba implementation issues. *IEE Colloquium (Digest)*, (007):5/1–5/3, January 14 1994.

[2] A. D. Birrell and B. J. Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1):39–59, February 1984.

[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] 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.

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

[6] 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.

[7] D. N. Jayasimha, S. S. Iyengar, and R. L. Kashyap. Information integration and synchronization in distributed sensor networks. *IEEE Trans. Syst., Man, Cybern.*, SMC-21(21):1032–1043, Sept./Oct. 1991.

[8] 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.

[9] A. Knoll and J. Meinkoehn. Data fusion using large multiagent networks: an analysis of network structure and performance. In *Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 113–120, Las Vegas, NV, Oct. 2-5 1994. IEEE.

[10] L. Lamport. Synchronizing time servers. Technical Report Technical Report 18, Digital System Research Center, 1987.

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

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

[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] L. Prasad, S. S. Iyengar, R. L. Kashyap, and R. N. Madan. Functional characterization of sensor integration in distributed sensor networks. *IEEE Trans. Syst., Man, Cybern.*, SMC-21, Sept./Oct. 1991.

[15] L. Prasad, S. S. Iyengar, and R. L. Rao. Fault-tolerant sensor integration using multiresolution decomposition. *Physical Review E*, 49(4):3452–3461, April 1994.

[16] T. Sundsted. An introduction to agents. *Java World*, June 1998.

[17] A. Watson. Omg (object management group) architecture and corba (common object request broker architecture) specification. *IEE Colloquium - Digest*, (007):4/1, January 4 1994.

[18] R. Wesson, F. Hayes-Roth, J. W. Burge, C. Stasz, and C. A. Sunshine. Network structures for distributed situation assessment. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):5–23, January 1981.

[19] 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.

Hairong Qi *received her Ph.D. degree in Computer Engineering from North Carolina State University in 1999, B.S. and M.S. degrees in Computer Science from Northern JiaoTong University in 1992 and 1995 respectively. She is now Assistant Professor in the Department of Electrical and Computer Engineering at the University of Tennessee, Knoxville. Qi is a member of IEEE and Sigma Xi. Her current research interests are multi-sensor data fusion, omni-directional sensor technology, knowledge discovery from database, and content-based image retrieval.*

S. Sitharama Iyengar *is the Chairman and Professor of Computer Science Department at Louisiana State University. He has been involved with research in high-performance algorithms, data structures, sensor fusion, data mining, and intelligent systems since receiving his Ph.D. degree (in 1974), his M.S. degree from the Indian Institute of Science (1970). He has served as a principal investigator on research projects supported by the Office of Naval Research, the National Aeronautics and Space Administration (NASA), the National Science Foundation, California Institute of Technology's Jet Propulsion Laboratory, the Department of Navy-NORDA, the Department of Energy, LEQFS-Board of Regents, and the U.S. Army Office. He authored and co-authored 11 books and over 270 research papers. Dr. Iyengar is a fellow of the IEEE and was the winner of the prestigious IEEE Technical Achievement Award for Outstanding Contributions to Data Structures and Algorithms in Image Processing and Sensor Fusion.*

Krishnendu Chakrabarty *received the B. Tech. degree from the Indian Institute of Technology, Kharagpur, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively, all in Computer Science and Engineering. He is now Assistant Professor of Electrical and Computer Engineering at Duke University. Dr. Chakrabarty is a recipient of the National Science Foundation's Early Faculty (CAREER) award, and a Mercator Professor award from the Deutsche Forschungsgemeinschaft, Germany for carrying out research at the University of Potsdam during 2000-2001. His current research projects (supported by NSF, DARPA and industrial sponsors) are in system-on-a-chip design & test, real-time operating systems, distributed sensor networks, thermal management in integrated circuits, and architectural optimization of microelectrofluidic systems. He has published over 50 papers in archival journals and refereed conference proceedings, and he holds a US patent in built-in self-test. He is a member of IEEE and Sigma Xi.*