Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 182561, 9 pages
doi:10.1155/2012/182561

Research Article

Cooperative Data Processing Algorithm Based on Mobile Agent in Wireless Sensor Networks

Shukui Zhang,[1,2] Yong Sun,[1] Jianxi Fan,[1] and He Huang[1]

[1] *School of Computer Science and Technology, Soochow University, Suzhou 215006, China*
[2] *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China*

Correspondence should be addressed to Shukui Zhang, zhangsk2000@163.com

Received 7 January 2012; Accepted 25 March 2012

Academic Editor: Hongli Xu

Mobile agent (MA) systems provide new capabilities for energy-efficient data processing by flexibly planning its itinerary for facilitating agent-based data collection and aggregation. In this paper, we present a cooperative data processing algorithm based on mobile agent (MA-CDP), and considers MA in multihop environments and can autonomously clone and migrate themselves in response to environmental changes. MA accounts for performing data processing and making data aggregation decisions at nodes rather than bringing data back to a central processor, and redundant sensory data will be eliminated. The results of our simulation show that MA-based cooperative data processing provides better performance than directed diffusion in terms of end-to-end delivery latency, packet delivery ratio, and energy consumption.

1. Introduction

The advances in Microelectromechanical System (MEMS) and wireless communication have enabled the development of a new kind of network—the wireless sensor network (WSN). One of the unique features of WSN applications is the necessity of cooperation. Each sensor node normally has limited sensing and processing capabilities, constrained power resources, and reduced communication bandwidth. Therefore, cooperation among sensor nodes is important in order to compensate for each other's capabilities as well as to improve the degree of fault tolerance, and the key to an effective cooperation is a combination of low-level sensor processing and local exchange of data to reach consensus in the neighborhood of the occurring event. This characteristic of WSNs brings up some important issues for cooperation communication, including energy efficiency, scalability, and reliability [1].

To address such challenges, most of researches focus on prolonging the network lifetime, allowing scalability for a large number of sensor nodes, or supporting fault tolerance (e.g., sensor's failure and battery depletion) [2, 3]. Most energy-efficient proposals are based on the traditional client/server computing model, where each sensor node sends its sensory data to a processing center or a sink node. Because the link bandwidth of a wireless sensor network is typically much lower than that of a wired network, a sensor network's data traffic may exceed the network capacity. To solve the problem of the overwhelming data traffic, Qi et al. [2] proposed the mobile-agent-(MA-) based distributed sensor network (MADSN) for collaborative signal and information processing.

Generally speaking, an MA is a special kind of software that can execute autonomously, with identification, itinerary, data space, and method as its attributes. An MA [4] is a computational process which has several characteristics: (1) "reactivity" (allowing agents to perceive and respond to a changing environment), (2) "social ability" (by which agents interact with other agents), and (3) "proactiveness" (through which agents behave in a goal-directed way). An MA may need to cooperate in order to achieve better and more accurate performance or need additional capabilities that it does not have. This cooperation takes place by doing a coalition formation which it is created by the fusion center agent. By cooperation we mean sharing data and resolving conflicts.

By transmitting the software code, namely, mobile agent (MA) to sensor nodes, the large amount of sensory data can be reduced or transformed into small data by eliminating

the redundancy. For example, the sensory data of two closely located sensors are likely to have redundant or common parts when the data of two sensors are merged. Therefore, data aggregation is a necessary function in densely populated sensor networks in order to reduce the sensory data traffic.

MA-based algorithm is a promising design paradigm that can be utilized to solve the overwhelming data traffic [5], especially over low bandwidth links, an MA selectively migrates among sensor nodes by moving the processing function to the target nodes, performs local processing by using resources available at the local nodes rather than bringing the data to a central processor (sink), and incrementally fuses the local decisions on each sensor node to reach a progressively accurate global decision.

This limitation is tackled by MADD algorithm [6]. The processes involved in MADD are divided into some sections. First, the MA is dispatched from sink to the first source node, and in the next place, the MA migrates from first source node to last source node, visiting selected source nodes in between. This algorithm does not always guarantee the best sequence of nodes to be visited.

In addition, node failures are a frequent occurrence in WSN. When a node fails, all data and MA residing on that node are permanently lost. While this may cause application failure, cooperative data processing provides the capability for applications to self-heal. Specifically, an application can heal itself by cloning or moving its agents onto the replacement node when it is installed. Unlike other in-network reprogramming systems, cooperative data processing enables application developers to control over the self-healing process. For example, in the fire tracking application, a node will fail when it catches on fire. The Fire Tracker agent heals itself by detecting this failure and cloning itself around the failed node to ensure the integrity of the perimeter.

Towards this end, we propose cooperative data processing based on MA algorithm (MA_CDP). With this algorithm, large amount of sensory data can be reduced or transformed into small data by eliminating the redundancy. Furthermore, to have better understanding in evaluating the performance of the algorithm, we present detailed analytical algorithm of data dissemination. With appropriate parameters set, the results of our simulation show that cooperative data processing provides better performance in terms of packet delivery ratio, energy consumption, and end-to-end delay.

2. Related Work

A traditional approach for WSN adaption is to reprogram it over the wireless network. Systems that enable this can be divided based on what is reprogrammed, that is, native code, interpreted code, or both. Two systems that reprogram native code are Deluge [7] and MOAP [8]. They are designed to transfer large program binaries, enable the network to be arbitrarily reprogrammed, but incur high overhead and latency. To address this, SOS [9], Contiki [10], and Impala [11] are systems that enable partial reprogramming of binary code by providing a microkernel that supports dynamically linked modules. Since modules are relatively small, the cost of reprogramming is lower.

Recently there has been a growing interest on the design, development, and deployment of MA systems for high-level inference in WSNs [12–15]. In [12], the agent design in WSNs is decomposed into four components, that is, architecture, itinerary planning, middleware system design, and agent cooperation. Among the four components, itinerary planning determines the order of source nodes to be visited during agent migration, which has a significant impact on energy performance of the MA system. It has been shown that finding an optimal itinerary is a NP-hard problem. Therefore, heuristic algorithms are generally used to compute competitive itineraries with a suboptimal performance.

In [13], two simple heuristics are proposed: (i) a local closest first scheme that searches for the next node with the shortest distance to the current node, and (ii) a global closest first scheme that searches for the next node closest to the dispatcher. These two schemes only consider the spatial distances between sensor nodes and, thus, may not be energy efficient in many cases.

Sharma and Mazumdar have investigated the use of limited infrastructure, that is, networks with a number of wired connections between sensor nodes, in [16]. Their approach establishes a small-world graph by utilizing wired links between a subset of nodes to reduce the overall energy demands as well as the different energy consumption rates of participating nodes. The additional efforts required for the wiring however make it suited for long-term deployments of sensor networks only.

Wagenknecht et al. also propose to deploy nodes with higher computational capabilities within a WSN to act as cluster-heads for sensor subnetworks, that is, partitions of the sensor network [17]. They use embedded systems with a 233 MHz clock frequency and 128 megabytes of RAM as the backbone to interconnect the sensor subnetworks through a wireless mesh network. Although deploying additional gateways allows for shorter multihop routes, the energy savings are possibly counterbalanced by the greater energy requirements of the gateways, which are not analyzed in detail in the paper.

A different approach to shift computational tasks into the network is the use of mobile agents. In such networks, data is not forwarded to an external sink, but instead, the processing application (the mobile agent), including its state variables, is sent to the node and executed locally [18]. As all process context data are contained within the agent, it can be supplied with input data at one node, while the processing can be performed at a different and more powerful system. We thus consider it a well-suited supplement to migrate tasks between nodes.

3. Collaborative Data Processing

An MA is a special process that can autonomously migrate across nodes. The migration transfers both the code and state, allowing the agent to resume execution at the destination. It is useful for performing computations that span

multiple nodes. When an agent migrates, it can either clone or move. If an agent is cloned, a copy of it arrives and starts executing at the destination while the original one resumes on the original node. If an agent is moved, it will no longer exist on the original node after it arrives at the destination. An agent's life cycle begins when it is either injected into the network from a base station or cloned from another agent already in the network. Each agent executes autonomously performing application-specific tasks, and multiple agents may reside on the same node. When an agent completes its tasks, it dies by freeing the computational resources which it used.

The order of source nodes to be visited by the MA can have a significant impact on energy consumption. Finding an optimal source-visiting sequence is an NP-complete problem [6]. In [19], a genetic algorithm-based solution to compute an approximate solution is presented. Though global optimization can be achieved by using genetic algorithm, it is not a lightweight solution for sensor nodes that are constrained in energy supply [20]. This paper adopts a gradient-based solution for the MA to dynamically decide the route.

3.1. Algorithm Overview. Figure 1 shows the sequence of operations to processes involved in WSN environment. When a sink receives a task request assigned by an application, the sink broadcasts the query packet. The query packet contains the sensing task description, interest region representation, along with other information. If the node finds it can satisfy the interest query, it declares itself as a source node. Each source node generates a response by exploratory data. Then, the sink receives a large number of exploratory data packets from various source nodes and decides the source-sequence list to be visited by MA. The MA-related operation begins at the point of the sink dispatching MA and ends when the MA returns to the sink with collected results. In most cases, each source is expected to generate the sensory data periodically with some interval, which means the same code (MA) needs to be stored for multiple running. Thus, when the MA arrives at the FirstNo, it will be stored. Then, it sets a Create-MA-Timer, which is used to trigger the next round to dispatch the MA to collect data from the relevant sources again. Obviously, the interval between the successive rounds will be equal to the sensory data generating rate which is set to the value of the Create-MA-Timer. This round will be repeated until the task is finished. A round can also be defined as the interval from the time during which an MA collects the data packet in the FirstNo to the time during which it collects the data packet in LastSrc. At the end of the last round, the task is finished.

When the MA arrives at the first source node, it continues visiting other source nodes until it reaches the last source node. Firstly, aggregated data is sent back to the sink along the reinforced path. Therefore, an MA reduces the amount of data to be transmitted via aggregating relevant data.

Consider an MA dispatched by the sink node to collect data from n source nodes. Let S_{code} be the size of the MA processing code, S_{head} the size of agent packet header, and S^0_{ma} the agent size when it is first dispatched by the sink node.

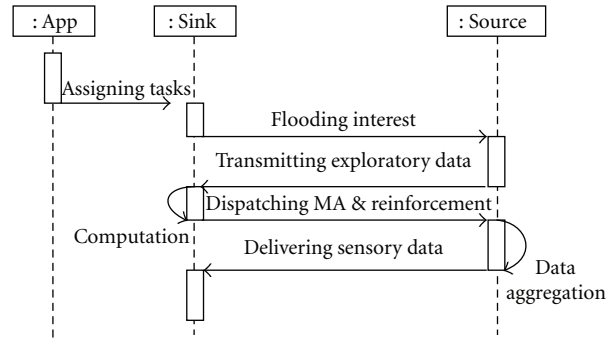

Figure 1: Routing sequence diagram.

Then we have $S^0_{\text{ma}} = S_{\text{code}} + S_{\text{head}} \cdot S_{\text{data}}$ is the size of raw data at a source node. The reduced data payload collected by the MA at each source is denoted as R_1, $R_1 = (1 - \rho) \times S_{\text{data}}$. where ρ is an aggregation ratio ($0 \leq \rho \leq 1$), a measure of the compression performance. Let S^k_{ma} be the MA size when it leaves the kth source ($1 \leq k \leq n$). Since there is no data aggregation at the first source, we have $S^1_{\text{ma}} = S^0_{\text{ma}} + R_1$.

Since the MA visits the second source node, it begins to perform data aggregation to reduce the redundancy between the data collected in the source and the data it carries. The MA size after it leaves the second source node is $S^2_{\text{ma}} = S^0_{\text{ma}} + R_1 + (1 - \rho)R_1$, and so forth. After visiting the kth source node, accumulated by MA can be calculated using the following formula (1):

$$\begin{aligned} S^k_{\text{ma}} &= S^{k-1}_{\text{ma}} + (1-\rho)R_1. \\ &= S^0_{\text{ma}} + [1 + (k-1)(1-\rho)]R_1. \end{aligned} \tag{1}$$

After visiting all the n source nodes, the MA has a size S^n_{ma} in the range $[S^0_{\text{ma}} + R_1, S^0_{\text{ma}} + n \times R_1]$. The lower bound $S^0_{\text{ma}} + S_{\text{re-data}}$ corresponds to a perfect aggregation model where multiple packets are compressed into a single one, while the upper bound $S^0_{\text{ma}} + n \times R_1$ corresponds to the case of no aggregation performed at the MA.

During the MA migration from one node to another, it aggregates sensory data and removes redundant data at the same time. The aggregated data can be calculated using the following equation:

$$S^i = \sum_{k=1}^{i} R_k. \tag{2}$$

Secondly, the energy cost is reduced. In client/server-based sensor network, all source nodes in target region transmit sensory data individually back to sink with a specific interval. In agent-based algorithm, the MA carries both the processing code as well as the source-visiting sequence.

3.2. MA Packet Format. The structure of MA packet is shown in Figure 2. The pair of SinkID and MA-SeqNum is used to identify an MA packet. Whenever a sink dispatches a new MA packet, it will increment the MA-SeqNum. FirstNo and LastNo are the source nodes scheduled to be visited firstly and lastly by the MA, respectively. The pair of FirstNo and

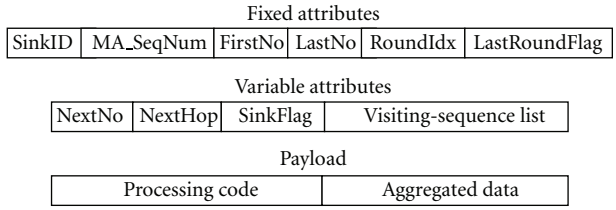

Figure 2: Structure of MA packet.

LastNo indicates the beginning and ending points of MA's data gathering. RoundIdx is the index of current round. The value is initially set to 1 by the sink in the first round and will be incremented by the FirstNo in the following rounds. Last Round Flag indicates that the current round is the last round of the whole task. The flag is set by FirstNo. When an MA with Last Round Flag set arrives at a source node, it can make the system unmount the corresponding processing code after its execution.

When an MA migrates, it may change variable attributes. NextNo specifies the next destination source node to be visited. NextHop indicates the immediate next hop node which is an intermediate sensor node or a target source node. If NextHop is equal to NextNo, it means that the next hop node is current destination source. Visiting-Sequence List contains the identifiers (IDs) of target sensor nodes that remain to be visited in the current round. It does not contain any information of source-visiting sequence since NextNo is dynamically decided when an MA arrives at a source node (except LastNo). Visiting-Sequence List initially contains all the IDs of source nodes when an MA is created. The corresponding ID will be deleted after the MA visits the source node. If all the target sources have been visited by the MA, Sink Flag is set to indicate that the destination of the MA is the sink. NextNo, NextHop, Visiting-Sequence List, and Sink Flag hint the dynamical route of MA migration. Payload includes two kinds of data. One is Processing Code which is used to process sensed data; the other is Aggregated Data which carries the accumulated data result. The size of Aggregated Data is zero when an MA is generated and increases while the MA migrates from source to source.

3.3. Cooperative MA Routing. The proposed MA_CDP mechanism is based on the original DD. In the DD, the sink initially diffuses an interest for notifications of low-rate exploratory events. Once target sources receive the corresponding interest, they send exploratory data, possibly along multiple paths, toward the sink. If the sink has multiple previous hop nodes, it chooses a preferred neighbor to receive subsequent data messages for the same interest (e.g., the one which delivered the exploratory data earliest). To do this, the sink reinforces the preferred neighbor, which in turn, reinforces its preferred previous hop node, and so on. Periodically, the source sends additional exploratory data messages to adjust gradients in the case of network changes (due to node failure, energy depletion, or mobility), temporary network partitions, or to recover from lost exploratory messages [21].

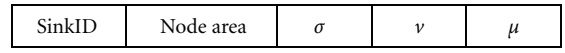

Figure 3: Instruction packet.

Suppose that the sensory data generating rate is ν, the aggregated data rate is μ, the number of transmitted aggregated data is u. To balance the energy consumption of the sensors in the sensing area, and to control the number of aggregated data sent to the sink in a distributed way, a variables σ is used, which is defined as $\sigma = u/N$. The sink calculates σ, and then sends the instruction (σ, ν, μ) back to the sensing area through the n_r relay nodes. The instruction is encapsulated with a packet header as shown in Figure 3.

We assume all sensors in the sensing area are well scheduled to wake up to receive the instruction from the sink for the next round of operation. However, the transmitted instructions are subject to packet losses in the WSN. If a sensor has not received the instruction by a predetermined time period, it sends a request through the n_r relay nodes to the sink and asks the sink to resend the instruction. Each sensor receiving the instruction creates a random number uniformly distributed in $[0, 1]$ and compares the random number with σ given in the instruction. If the random number is larger than σ, the sensor turns to sleep, otherwise it turns to sample the source signal and quantize its readings with ν bits in the next round of operation. If the random number of a sensor is larger than σ, the sensor is also selected to aggregated data with aggregated data rate μ and sends its aggregated data to the sink.

Since the ultimate goal is the detection of events in sensor networks [22], the sink may stop handling any exploratory message flows if it considers that the number of source nodes is large enough to meet the requirement of reliable event detection. Thus all the source nodes or only a subset of these nodes will be chosen to be visited by MA. Among the target source nodes to be visited, the sink will choose the first and last source nodes. Then, the sink generates an MA with the packet format described in Figure 4 and dispatches it to the first source. At the same time, the sink reinforces the path to the last source. When the MA arrives at the first source node, it is stored in the node. We divide the whole task period into rounds, where each round requires the MA to visit all the chosen target sensors and to return the data result to the sink. The MA starts from the first source (or from the sink only in the first round) and arrives at the last source. Finally, the MA will carry the data results to the sink along the reinforced path. In the first round, in addition to that the MA moves from source to source to collect and aggregate information, it also copies processing code into the memory of each source node. At the beginning of each round, the first source node will construct another MA from its memory and dispatch it to initiate the new round. Since processing code has already resided in each source node after the first round, the MA does not carry the processing code any more in the following rounds. When the whole task is finished, all the source nodes will discard the processing code.

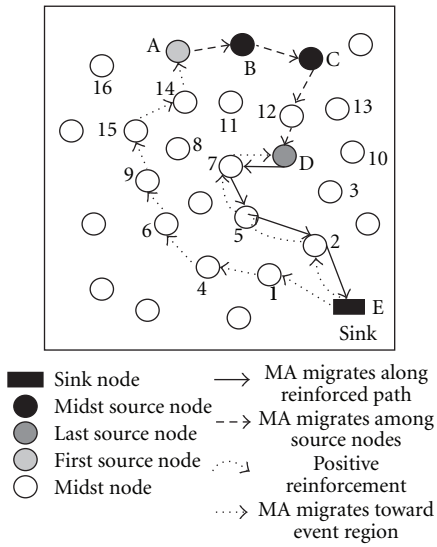

Figure 4: Cooperative MA routing.

In MA_CDP, target source nodes flooding exploratory messages enable sensor nodes to set up ToSourceEntry, which is a kind of gradient toward each target source. ToSourceEntry is used for MA to roam among source nodes. In this paper, a time-to-live (TTL) field is set in exploratory message to mandate only the sensor nodes within the target region to set up their ToSourceEntries. The value of TTL is decreased as exploratory message is propagated hop by hop. If the value is equal to 0, sensor nodes do not set up ToSourceEntry anymore. Among all the neighbors of a sensor node, only the neighbor who first relays the exploratory message of a specific target source will be chosen as the sensor node's NextHop in the ToSourceEntry. In Figure 4, nodes A, B, C, and D are the target source nodes. The ToSourceEntries set up by nodes A, B, C, 12, and D are shown in Table 1.

Based on the gradients and ToSourceEntries, a migrating route is decided by the following three operating elements.

(1) Choose *FirstNo* and *LastSrc*. According to (1), the size of an MA is the minimum in *FirstNo* while it becomes the maximum in *LastSrc*. Thus, to reduce total communication overhead, *FirstNo* should be the farthest target sensor from the sink, while *LastNo* should be the closest one. In this paper, the target source which is the last (first) to send exploratory messages to the sink is chosen as *FirstNo*(*LastSrc*). The sink will reinforce the path to *LastSrc*.

(2) Decide Source-Visiting Sequence. Except that *FirstNo* and *LastNo* are chosen by the sink, the sequence of visiting the other source nodes is dynamically decided by each target sensor in *Visiting-Sequence List*. For example, when an MA arrives at node A in Figure 4, the node will choose the closest next source node based on its ToSourceEntry shown in the first row of Table 1. Since the lowest latency of node B is the least, it implies that node B is the closest source node from node A and is chosen as *NextNo*.

(3) Find the next hop node to route an MA along the entire path from sink to source, source to source, and source to sink. Dispatched by the sink, an MA migrates to *FirstNo* in the same manner as a reinforcement message is forwarded in original DD. When the MA migrates among target sources, its next hop node will be decided according to current node's ToSourceEntry. The MA will return to the sink using the reinforced path (e.g., path D-7-5-2-E in Figure 2).

4. Performance Analysis

There are n nodes $S_k(1 \le k \le n)$ in the network. Since the transmission of instruction and request packets are to ensure all the sensors in the sensing area receive the instruction, the average number of instruction packets sent by the sink in an operation round is determined by the following analysis.

Without loss of generality, we assume that m among n_s aggregated data are received by the sink. We assume the packet loss rate per hop in the wireless channels is uniformly distributed with mean of φ. The probability of receiving v quantized readings is

$$p_m = C_{n_s}^{m}\left(1 - p_\rho\right)^m p_\rho^{n_s - m}, \tag{3}$$

where p_ρ denotes the end-to-end packet loss rate from the sensors in the sensing area to the sink and is

$$p_\rho = 1 - (1 - \varphi)^{n_s + 1}. \tag{4}$$

The probability that after i multicasts the instruction packet sent from the 1st relay node has been successfully received by all the N nodes in the sensing area is given by $(1 - \varphi^i)^N$. The probability that the instruction packet is successfully received by all the N nodes after exactly i multicasts is then given by

$$p_i = \left(1 - \varphi^i\right)^N - \left(1 - \varphi^{i-1}\right)^N. \tag{5}$$

Suppose after the sink sends exactly n $(n \geqslant i)$ instruction packets, all the N nodes successfully receive the instruction. This means the last transmission of instruction packet from the sink (through n_r hops) to the 1st relay node is successful and the nodes in the sensing area which are waiting for the instruction finally receive it. The probability that i transmissions (including the last successful transmission) among the n transmissions to the 1st relay node are successful is therefore given by

$$\begin{aligned}\varphi(n,i) &= C_{n-1}^{i-1}\left[1 - (1-\varphi)^{n_r}\right]^{n-i} \times \left[(1-\varphi)^{n_r}\right]^{i-1}(1-\varphi)^{n_r} \\ &= C_{n-1}^{i-1}\left[1 - (1-\varphi)^{n_r}\right]^{n-i}(1-\varphi)^{n_r}.\end{aligned} \tag{6}$$

Table 1: ToSourceEntry setup after exploratory messages flooding.

ToSourceEntry (SeqNum = 5)								
	A		B		C		D	
	Next hop	Cast (ms)	Next hop	Cast (ms)	Next hop	Cast (ms)	Next hop	Cast (ms)
A	—	—	B	4.46	B	8.24	B	16.32
B	A	4.47	—	—	C	4.43	C	12.89
C	B	8.16	B	4.32	—	—	12	8.52
12	C	9.65	C	7.56	C	4.86	D	5.08
D	12	14.15	12	12.67	12	8.73	—	—

Considering the joint distribution of p_i and $\varphi(n, i)$, the probability that all the N nodes receive the instruction after the sink sends exactly n instruction packets is given by

$$\begin{aligned}\varphi(n) &= \sum_{i=1}^{n}\varphi(n,i)\times p_i \\ &= \sum_{i=1}^{n} C_{n-1}^{i-1}\left[1-(1-\varphi)^{n_r}\right]^{n-i}(1-\varphi)^{n_r} \\ &\quad\times\left[\left(1-\varphi^{i}\right)^{N}-\left(1-\varphi^{i-1}\right)^{N}\right].\end{aligned} \tag{7}$$

The average number of instruction packets sent by the sink is then given by

$$N_1 = \sum_{n=1}^{\infty} n\varphi(n). \tag{8}$$

Since a request packet is sent from the sensing area to the 1st relay node if there is at least one node that does not receive the instruction, the fact that there are n instruction packets sent by the sink means there are $(n-1)$ request packets sent from the sensing area. The average number of request packets sent from the sensing area is therefore

$$N_R = N_1 - 1. \tag{9}$$

In addition, we assume that each node has a probability P_k of being able to successfully complete the agent's task and energy consumption E_k required for the agent to process the data at node k. The energy consumption for the agent to move between nodes is given by E_{kj}. MA_CDP is to minimize the expected energy consumption and the expected time (in terms of the number of hops) to successfully complete the task.

Without loss of generality, we make several assumptions to simplify the MA_CDP model.

(1) The target is stationary. Therefore, $Z_k(t)$, the measurement of node k at time t, remains constant during the MA migration process.

(2) $E_{kj} = E$ for all the migration steps.

(3) $E_k = e$ for all sensor nodes.

The expected energy consumption to complete the task or visit all nodes in failure for a route $R = \langle S_1, S_2, \ldots, S_n\rangle$ is

$$\begin{aligned}E_R &= E + e + p_1E + \sum_{i=2}^{n}\left(\prod_{j-1}^{i-1}\left(1-p_j\right)(E+e+P_iE)\right) \\ &\quad+\prod_{j=1}^{n}\left(1-p_j\right)E.\end{aligned} \tag{10}$$

The equation can be explained as follows. The first site, S_1 is always visited and consumes E amount of energy. Upon arrival, energy e must be spent regardless of success or failure. With probability p_1, the task is successfully completed and the agent can return to node 0 with energy cost of another E. However, with probability $(1 - p_1)$, that is, the failure rate, the agent migrates to node S_2. The expected energy consumption used by the MA moving from node S_1 to S_2 is $(1 - p_1)E$. Similarly, the MA consumes energy E to migrate from node $i - 1$ to node i, and then with probability p_i, it succeeds at node i and returns to node 0 consuming energy E. Hence, the accumulated energy consumption at node S_i is $\prod_{j-1}^{i-1}(1 - p_i)(E + e + p_iE)$. Finally, the last round arises when failure occurs at all nodes and the agent must return to the originating node 0 with an energy consumption E. Furthermore, the expected number of hops to complete the task or visit all nodes in failure, for a route $R = \langle S_1, S_2, \ldots, S_n\rangle$, is

$$\begin{aligned}H_{\mathrm{op}R} &= 1\cdot P_1 + \sum_{i=2}^{n}\left(i\cdot\prod_{j=1}^{i-1}\left(1-p_j\right)p_i\right) \\ &\quad+(n+1)\prod_{j=1}^{n}\left(1-p_j\right).\end{aligned} \tag{11}$$

We use the following equation to model the probability of success:

$$p_k = 1 - \frac{\left(D_{\mathrm{esire}} - \sum_{i=0}^{\mathrm{hop}} I_i\right)}{I_{\max}}, \tag{12}$$

where $I_{\max}$ is the maximum information gain a sensor node can provide. H_{op} is the total node numbers the MA has migrated through. We then have the following theorem.

Theorem 1. *The optimal route for MA_CDP is attained if the nodes are visited in the decreasing order of I_k, $k = 1, 2, \ldots n$, that is, $I_1 > \cdots > I_k > \cdots > I_n$.*

Proof. We employ a similar proof method in [20]. Consider the effect of switching the order of two adjacent nodes on the route, say k and $k+1$. We call this new route as R'; only the kth and $(k+1)$st terms are affected by the switch. The terms appearing before the kth term do not contain anything involving k or $k+1$. Terms that follow the $(k+1)$st term, on the other hand, all contain $(1-p_k)(1-p_{k+1})$ in the same way. Then the difference in the expected energy consumption is

$$E_R - E_{R'} = (E+e)\prod_{j-1}^{k-1}\left(1-p_j\right)(p_{k+1}-p_k), \tag{13}$$

and the difference in the expected number of hops is

$$H_{\mathrm{op}R} - H_{\mathrm{op}R'} = \prod_{j-1}^{k-1}\left(1-p_j\right)(p_{k+1}-p_k). \tag{14}$$

Since $p_k > p_{k+1}$, R is a better route with a smaller expected energy consumption and number of hops.

This indicates that when the kth node on the route has a smaller probability than the $(k+1)$st node in making the agent complete its job, then we can decrease the expected energy consumption and number of hops by switching them.

From (6), we find that the probability of success on a sensor node is directly related to the total information utility the MA accumulates. The higher the total information gain the MA carries, the more likely the agent will finish the task on the current sensor node—thus the higher the probability will be. So the optimal route for MA_CDP is the sequence with decreasing information gain. □

Theorem 2. *If $p_k = p$ for all sensor nodes, then the expected number of hops $H_{\mathrm{op}R}$ algorithm 1/p as the number of sensor nodes n increases.*

Proof.

$$\begin{aligned} H_{\mathrm{op}R} &= 1\cdot p + \lim_{x\to\infty}\left[\sum_{i=2}^{n}\left(i\cdot\prod_{j=1}^{i-1}\left(1-p_j\right)\right)P \right. \\ &\qquad \left. +(n+1)\prod_{j=1}^{n}(1-p)\right] \\ &= \sum_{i=1}^{\infty} i(1-p)^{(i-1)}P + \lim_{x\to\infty}(n+1)\prod_{j=1}^{n}(1-p) \\ &= \frac{1}{p}. \end{aligned} \tag{15}$$

□

This theorem shows that we need to improve the probability of success on each sensor node in order to reduce the number of hops for the MA to finish the task.

5. Simulation Experiments and Evaluation

In order to demonstrate the performance of MA_CDP, we choose a client/server-based scheme (i.e., DD) to compare with MA_CDP. We use NS2 for discrete event. Each task requires periodic transmission of data packets with a constant bit rate (CBR) of 1 packet/s. We assume that both the sink and sensor nodes are stationary. The parameter values used in the simulations are presented in Table 2. The basic settings are common to all the experiments. For each experiment, we simulate for sixty times with different random seeds and get the average results.

Table 2: Simulation setting.

Basic specification	
Network size	500 m∗500 m
Topology mode	Randomized
Total sensor node number	1500
Data rate MAC layer	1 Mbps
Transmission range of sensor node	60 m
Sensed data packet interval	1 s

Three performance metrics are evaluated: *packet*, *delivery,* and *ratio.* It is denoted by *u.* It is the ratio of the number of data packets delivered to the sink to the number of packets generated by the source nodes. *Energy consumption per successful data delivery* It is denoted by *e*. It is the ratio of network energy consumption to the number of data packets successfully delivered to the sink. Let E_{total} be all the energy consumption by transmitting, receiving, and processing during simulation. n_{data} denotes the number of data packets delivered to the sink. Then, $e = E_{\mathrm{total}}/n_{\mathrm{data}}$. *Average end-to-end packet delay* is denoted by T_{ete}. And we also use T_{dd} and T_{ma} to denote the average end-to-end delays in DD and MA_CDP, respectively.

Though these conditions are affected by many parameters, only a set of important parameters is chosen, such as Default, for all sensor nodes has a probability P of being able to successfully complete the agent's task, $P = 0.6$, the duration of the task (T_{task}), data reduction ratio ($r = 0.8$), MA accessing delay ($\tau = 9$ ms), aggregation ratio ($q = 0.2$), size of sensed data of each sensor ($S_{\mathrm{data}} = 1$ KB). If we set q to 0, it means that data aggregation does not work; all the reduced sensed data are concatenated. Only one parameter (e.g., T_{task}, r, q, and S_{data}) is changed in each group while the other parameters are fixed. Several groups of simulations are evaluated.

In Figure 5, in these experiments, we change T_{task} from 10 seconds to 600 seconds, e decreases as T_{task} increases. When the T_{task} is small (i.e., lower than 60 seconds), MA_CDP has higher e than DD because MA_CDP consumes energy (E_q) to transmit processing code from the sink to the target region. If T_{task} is small, n_{data} is small, and e is large. However, when T_{task} is beyond 100 seconds with r equal to 0.8 and q equal to 0.2, MA_CDP has lower e than DD. Thus, to amortize the cost of shipping the processing code once to source node, the source should process enough long streams of data.

In Figure 6, in these experiments, we change MA accessing delay (τ) from 0 seconds to 0.05 seconds, and T_{dd} is constant since changing τ has no effect on DD. Since the

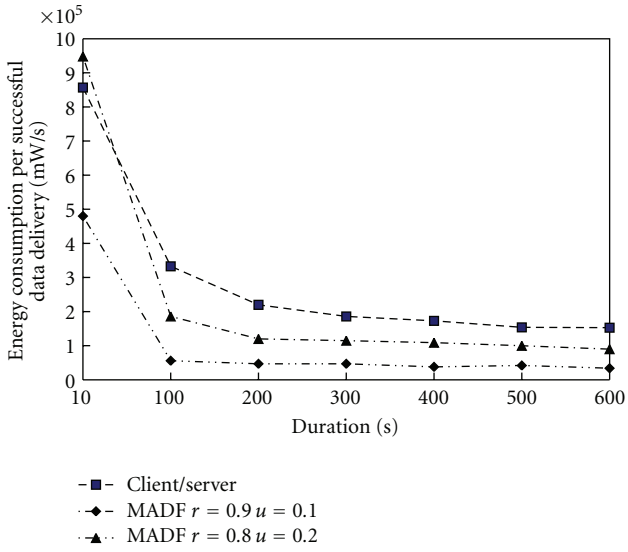

Figure 5: The impact of T_{task} on e.

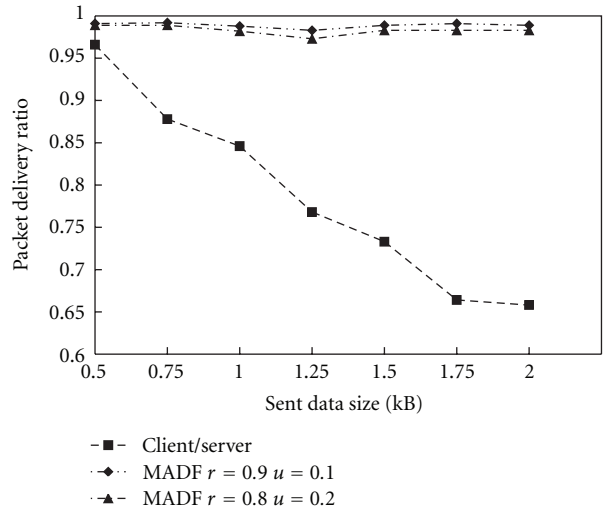

Figure 7: The impact of S_{data}, r, q on u.

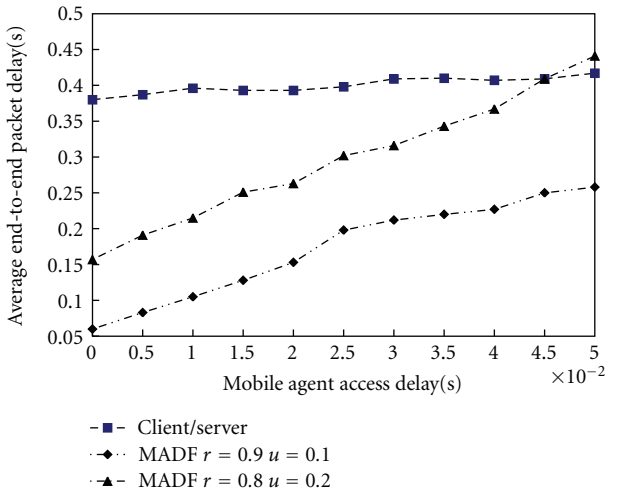

Figure 6: The impact of τ on T_{ete}.

delay of τ is introduced when MA visits each source, τ causes T_{ma}, increase fast if the value is set to a large value. When τ is beyond 0.042 seconds with r equal to 0.8 and q equal to 0.2, MA_CDP has larger end-to-end delay than DD. The value of τ is dependent on the middleware environments of MA system.

In Figure 7, in these experiments, we change the size of sensed data of each sensor (S_{data}) from 0.5 KB to 2 KB by increasing 0.25 KB each time and keep the other parameters unchanged. For MA_CDP, several groups of simulations are evaluated with variables r and q. In Figure 7, MA_CDP always outperforms DD in terms of q. In MA_CDP, only single data flow is sent for each round. In contrast, multiple data flows from individual source nodes are sent in DD. Thus, congestion in DD is more likely to happen than in MA_CDP. When S_{data} increases, the congestion is more serious and q of DD will decrease more.

In Figures 5, 6, and 7, MA-CDP exhibits more consistent and relatively higher reliability, lower energy consumption than DD by compromising end-to-end delay bound possibly in most scenarios. These figures also give hints that MA_CDP should choose r and q appropriately. It can be observed that MA_CDP has no advantage if q is equal to 0.2 and r is smaller than 0.4. Note that the value of r and q is dependent on the type of application. Before we adopt MA-CDP for data dissemination, the features of the application should be investigated. MA_CDP will be selected if enough high r and/or q can be attained.

6. Conclusion

In the environments where the source nodes are close to one another and generate a lot of sensory data traffic with redundancy, transmitting all sensory data by individual nodes not only wastes the scarce wireless bandwidth, but also consumes a lot of battery energy. Recently, MA-based distributed sensor network for collaborative signal and information processing is proposed as a solution to overcome these problems. In this paper, we addressed the problem of optimized itinerary planning for MAs in dense WSNs. Based on a general data aggregation model, we presented a cooperative data processing based on mobile agent algorithm (MA-CDP) and considered that MA dynamically enter a network and can autonomously clone and migrate themselves in response to environmental changes. MA-CDP routing scheme is proposed for MA to efficiently migrate from sink to source, source to source, and source to sink. We showed that the proposed schemes achieve considerable improvements in energy savings, packet delivery ratio, and end-to-end delivery delay.

Disclosure

The material in this paper was presented in part at The 4th International Conference on Wireless Communications, Networking and Mobile Computing,October, 2008 Dalian, China.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grants nos. 61070169, 61170021 and Natural Science Foundation of Jiangsu Province under Grant no. BK2011376 Specialized Research Foundation for the Doctoral Program of Higher Education of China no. 20103201110018, and Application Foundation Research of Suzhou of China no. SYG201034, SYG201118, and sponsored by Qing Lan Project.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–105, 2002.

[2] H. Qi, Y. Xu, and X. Wang, "Mobile-agent-based collaborative signal and information processing in sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1172–1183, 2003.

[3] J. J. Chang, P. C. Hsiu, and T. W. Kuo, "Search-oriented deployment strategies for wireless sensor networks," in *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC '07)*, pp. 164–171, Santorini Island, Greece, May 2007.

[4] M. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.

[5] I. Joe, "A path selection algorithm with energy efficiency for wireless sensor networks," in *Proceedings of the 5th ACIS International Conference on Software Engineering Research, Management, and Applications (SERA '07)*, pp. 419–423, August 2007.

[6] M. Chen, K. Taekyoung, and C. Yanghee, "Data dissemination based on mobile agent in wireless sensor networks," in *Proceedings of the IEEE Conference on Local Computer Networks (LCN '05)*, pp. 527–528, Sydney, Australia, November 2005.

[7] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys" 04)*, pp. 81–94, ACM, November 2004.

[8] T. Stathopoulos, J. Heidemann, and D. Estrin, "A remote code update mechanism for wireless sensor networks," Tech. Rep. CENS-TR-30, UCLA, 2003.

[9] C. C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava, "A dynamic operating system for sensor nodes," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services (MobiSys '05)*, pp. 163–176, ACM, June 2005.

[10] A. Dunkels, B. Gronvall, and T. Voigt, "A lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN '04)*, pp. 455–462, IEEE Computer Society, 2004.

[11] T. Liu and M. Martonosi, "Impala: A middleware system for managing autonomic, parallel sensor systems," in *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 107–118, ACM, June 2003.

[12] M. Chen, S. Gonzalez, and V. C. M. Leung, "Applications and design issues for mobile agents in wireless sensor networks," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 20–26, 2007.

[13] H. Qi and F. Wang, "Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '01)*, Helsinki, Finland, June 2001.

[14] M. Chen, T. Kwon, Y. Yuan, Y. Choi, and V. C. M. Leung, "Mobile agent-based directed diffusion in wireless sensor networks," *Eurasip Journal on Advances in Signal Processing*, vol. 2007, Article ID 36871, 13 pages, 2007.

[15] 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.

[16] G. Sharma and R. Mazumdar, "Hybrid sensor networks: a small world," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC '05)*, pp. 366–377, usa, May 2005.

[17] G. Wagenknecht, M. Anwander, T. Braun, T. Staub, J. Matheka, and S. Morgenthaler, "MARWIS: a management architecture for heterogeneous wireless sensor networks," in *Proceedings of the 6th International Conference on Wired/Wireless Internet Communications (WWIC '08)*, 2008.

[18] X. Wang, A. Jiang, and S. Wang, *Advances in Intelligent Computing*, vol. 3645/2005, Springer, Berlin, Germany, 2005.

[19] M. G. Lee and S. Lee, "Data dissemination for wireless sensor networks," in *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC '07)*, pp. 172–179, Santorini-Island, Greece, May 2007.

[20] W. Choi and S. K. Das, "A novel framework for energy-conserving data gathering in wireless sensor networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 1985–1996, Miami, Fla, USA, 2005.

[21] Q. Wu, N. S. V. Rao, J. Barhen et al., "On computing mobile agent routes for data fusion in distributed sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 6, pp. 740–753, 2004.

[22] H. Chen, H. Mineno, and T. Mizuno, "An energy-aware routing scheme with node relay willingness in wireless sensor networks," in *Proceedings of the 1st International Conference on Innovative Computing, Information and Control (ICICIC '06)*, pp. 397–400, Beijing, China, August 2006.