J Inf Process Syst, Vol.11, No.1, pp.148~164, March 2015
http://dx.doi.org/10.3745/JIPS.03.0018

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

# An Energy Efficient Distributed Approach-Based Agent Migration Scheme for Data Aggregation in Wireless Sensor Networks

Govind P. Gupta*, Manoj Misra*, and Kumkum Garg**

**Abstract**

The use of mobile agents for collaborative processing in wireless sensor network has gained considerable attention. This is when mobile agents ar

e used for data aggregation to exploit redundant and correlated data. The efficiency of agent-based data aggregation depends on the agent migration scheme. However, in general, most of the proposed schemes are centralized approach-based schemes where the sink node determines the migration paths for the agents before dispatching them in the sensor network. The main limitations with such schemes are that they need global network topology information for deriving the migration paths of the agents, which incurs additional communication overhead, since each node has a very limited communication range. In addition, a centralized approach does not provide fault tolerant and adaptive migration paths. In order to solve such problems, we have proposed a distributed approach-based scheme for determining the migration path of the agents where at each hop, the local information is used to decide the migration of the agents. In addition, we also propose a local repair mechanism for dealing with the faulty nodes. The simulation results show that the proposed scheme performs better than existing schemes in the presence of faulty nodes within the networks, and manages to report the aggregated data to the sink faster.

**Keywords**

Agent Migration Protocol, Data Aggregation, Mobile Agent, WSN

## 1. Introduction

The recent advancements in micro-electro-mechanical systems (MEMSs) and wireless communication technologies make it possible to build a large scale wireless sensor network (WSN) with a set of hundreds or more sensor nodes [1]. Nowadays, WSNs are used in a wide range of applications, including battlefields and border surveillance, environment and habitat monitoring, industrial process monitoring and control, health-care assistance, home automation, automatic target detection and tracking, etc. [1,2]. Some of the discussed WSN applications requires the remote retrieval of sensor data


**Manuscript received:** June 19, 2013; accepted August 20, 2013; onlinefirst October 28, 2014.
**Corresponding Author:** Govind P. Gupta (gpgupta3@gmail.com)
* Department of Computer Science and Engineering, Indian Institute of Technology, Roorkee 247667, India (gpgupta3@gmail.com, manojfec@iitr.ac.in)
** Department of Computer Science & Engineering, Manipal University, Jaipur 302026, India (kgargfec@gmail.com)

and are known to be data intensive [2]. Over a decade, the mobile agent computing paradigm has been employed for the remote retrieval of sensor data from a WSN where a set of agents are used to reduce the communication overhead by moving the processing code to the data source, rather than bringing the data to a central processing element (PE) or a sink [3,4]. This approach is a promising data retrieval technique that can be utilized to solve the overwhelming data traffic in WSN by reducing the inherent redundancy of raw data [3,4].

A mobile agent is an autonomous software entity that has the ability to migrate along its itinerary (i.e., migration path) and progressively aggregate the sensor data at each host node [2]. In this paper, we have used a migration path and itinerary synonymously. The aggregated data is then carried by the agent to the next node, where the processing code of the agent starts again and aggregation is performed upon the new data, as well as on the previous data. By dispatching the agents to sensor nodes, a large amount of sensor data may be filtered at each node by eliminating data redundancy [2-4]. The efficiency of a mobile agent-based data aggregation and collection scheme depends upon the agent migration scheme, which affects the overall energy consumption and aggregation cost [5]. The solutions that have been proposed until now, are either using a centralized approach [6-14] or a dynamic and distributed approach [15-18] to determine the migration paths of the agents. In a dynamic and distributed approach, the migration path of an agent is determined using local information at each host, while in a centralized approach, it is pre-computed at the sink node before an agent is dispatched and it is based on the global information of the network topology.

Recently, Mpitziopoulos et al. [2] proposed a clone-based itinerary design (CBID) scheme, which from among all of the proposed centralized approach based migration schemes, this scheme has been proven to be efficient in terms of data aggregation cost and overall response time. In CBID, all the sensor nodes need to transfer their neighbors' location information to the PE or the sink in order to maintain the global information of the network topology. Since WSN topology changes due to node failures or interference, this enforces a periodic update of global information of network topology. Repeating this process periodically brings a high communication overhead to the sensor nodes. Thus, the energy levels of the sensor nodes may drain quickly. To overcome this problem, we are proposing a dynamic and distributed version of CBID [2], called a clone-based dynamic and distributed agent migration (CDDAM) scheme. Our scheme is where the sensor nodes are organized in the rooted spanning trees, $T_i$, in a distributed fashion and where multiple agents are dispatched by the PE, one for each root of spanning trees of $T_i$. When a mobile agent $MA_j$ visits a sensor node $SN_i$ of tree $T_i$ with two or more children, the agent builds a clone of itself $MA_{clone_{j,k}}$ similar to [2], where $k = 1, 2 \ldots m$ and $m$ is the number of children of the visiting sensor node $SN_i$. $MA_{clone_{j,k}}$ is called the slave agents (SAs) and each SA visits a branch of the tree rooted at $SN_i$. When all mobile agent clones $MA_{clone_{j,k}}$ return to the root node $SN_i$, they handover the collected data to the mobile agent, $MA_j$.

We used three metrics, which are called the average energy consumption, the overall response time, and the success rate of agent's trip, to evaluate the performance of the CDDAM scheme in comparison with the previous schemes of CBID [2], TBID [14], and MMADIDD [19]. The CBID [2] and TBID [14] schemes are well known centralized approach-based agent migration schemes. However, MMADIDD [19] is a distributed approach-based agent migration scheme. The simulation results show that CDDAM performs better than previous schemes in the presence of faulty nodes within networks, and manages to report aggregated data to the sink faster.

The rest of this paper is organized as follows: in Section 2, we briefly describe other related work. In Section 3, we present the system model and list the assumptions contained in our work. In Section 4, we describe the CDDAM protocol in detail. In Section 5, we describe simulation scenarios and discuss the performance analysis of the protocols with respect to the selected metrics of interest. Finally, we conclude this paper in Section 6.

## 2. Related Work

Over the last decade, the agent-based data aggregation schemes have been proposed for energy efficient and scalable data collection operations in WSNs, where a mobile agent visits a set of nodes and progressively aggregates the data and returns back to the sink with the aggregated results. The efficiency of agent-based data aggregation depends on its migration protocols. In this section, we present a review study on the existing agent migration schemes that can be classified as a centralized or distributed approach-based scheme, as shown in Fig. 1. This classification is based on the place where agents' migration decisions are made. Table 1 shows the comparison of agent migration protocols that have been proposed up until now.

### 2.1 Centralized Approach-Based Schemes

In centralized approach-based schemes, the sink or PE collects the global information of network topology and computes the optimal migration path of the agents before dispatching them for data aggregation. In [6,7], the authors proposed a mobile agent-based distributed sensor (MADSA) network for energy efficient and scalable data aggregation. In [6], performance improvement of MADSA over the client/server model is shown using both analytically and through simulations. In [7], the authors proposed two simple heuristic algorithms, local closest first (LCF) and global closest first (GCF), to determine the itinerary for an agent migration. These heuristic algorithms are centralized and are not scalable since they are based on the global information of the network topology.

Wu et al. [8] proposed a genetic algorithm-based solution for planning the itinerary of an agent. This algorithm provides superior performance than the LCF and GCF algorithms do. However, the algorithms proposed in both [7] and [8] employ a single agent for data aggregation and collection tasks. The performance of a single agent-based approach declines as the network size increases. This is due to an increase in the agent's state size as it traverses more sensor nodes, which results in more energy consumption and round trip delays. The algorithms proposed in [7,8] assume a perfect data aggregation model, which is a non-realistic assumption.

Mpitziopoulos et al. [9] have proposed a near-optimal itinerary design (NOID) algorithm where the Esau-Williams heuristic is used for finding an appropriate number of mobile agents and their near-optimal itineraries. In NOID, the parallel deployment of multiple agents is suggested where each agent visits a subset of nodes. NOID outperforms the single agent-based approaches (e.g., LCF, GCF, and GA), in terms of data fusion cost and overall response time [9], but it endures high computational complexity in determining the agents' itineraries.

In [10-13], the authors proposed multi-agent itinerary planning (MIP) algorithms that help in the

collection of concurrent sensor data to reduce latency. These algorithms differ in source node grouping methods. In [12], the authors used an angle gap for grouping all the source nodes in a particular direction as a single group. This approach does not describe how to find out an optimal angle gap threshold. In [13], the authors proposed a genetic algorithm by encoding how many agents are dispatched and which sensor nodes are visited by individual agents. The limitation of a genetic algorithm-based approach is its higher computational complexity [20]. These algorithms assume that the set of source nodes to be visited by the agents are predetermined, which limits the application scope of the network.

Mpitziopoulos et al. [2] proposed a CBID algorithm where agents are dispatched in parallel that sequentially visit nodes arranged in tree structures and after visiting a node with two or more child nodes, the master agent makes clones of themselves. Each slave agent visits a sub-branch of the tree. When all slave agents return back to their parent node, they hand over their collected data to the master agent [2]. In CBID [2], agent's packet needs to carry extra cloning information about where to make clone. This algorithm has limited scalability.

In [14], the authors proposed a greedy tree-based itinerary design (TBID) algorithm to find near optimal itineraries for multiple agents. This algorithm is executed centrally at the sink and statically determines the number of agents that should be employed and their itineraries. The main theme of the TBID algorithm is to divide the area around the sink into concentric zones to construct the near optimal itinerary tree from inner zones to the outer zones. They used post order traversal with a possible shortcutting of the itinerary tree to derive an itinerary for each agent.

The main limitation of a centralized approach-based agent migration scheme is that it uses static migration paths (i.e., itineraries), which is based on a stale view of the network topology. This approach lacks dynamic recovery from node or communication link failures.

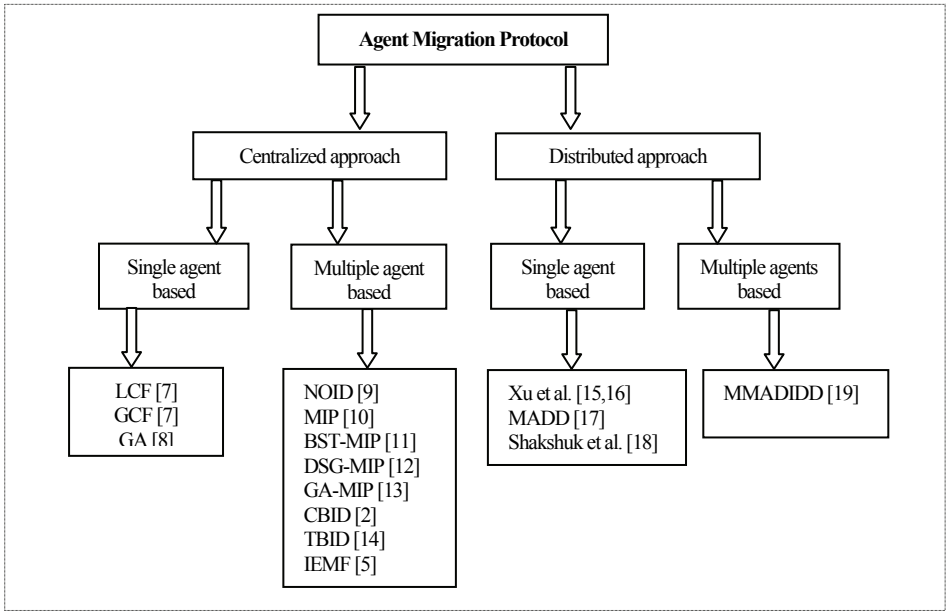



**Fig. 1.** Classification of agent migration protocols for data aggregation in wireless sensor networks

## 2.2 Distributed Approach-Based Schemes

The distributed approach-based agent migration protocol helps the agent to change the route dynamically according to the current state of the network.

Xu and Qi [15,16] proposed a dynamic agent migration algorithm for a target tracking application. In this method, an agent migrates to a sensor node that can get more accurate information about the target location by consuming less energy. For selecting the next node, they defined a cost function, which includes the following three components: energy consumption, information gain, and the remaining energy of a node. Once an agent accumulates sufficient information so that the accuracy of the estimation meets the desired level, the agent will terminate the migration and return to the sink [15,16]. This algorithm is more time expensive and may face difficulty in returning back to the sink without additional forwarding information.

In [17], the authors proposed the mobile agent-based directed diffusion (MADD) where an agent visits a subset of nodes. In MADD, the sink uses the first phase of the directed diffusion algorithm [21] to determine the subset of nodes. However, the actual data aggregation is carried out by dispatching an agent that sequentially visits the subset of nodes [20]. Shakshuki et al. [18] proposed a software agent-based directed diffusion where the order of node visits is determined at the sink node, but the authors did not describe the procedure. This method takes the routing cost and the remaining energy of a node for selecting a next node to be visited by an agent. The main limitations of the schemes described in [17,18] are that they depend on a directed diffusion scheme, that they incur extra communication overhead for agent migration, and that they are only applicable for query based data gathering applications.

**Table 1.** Comparison of agent migration schemes

| Scheme | Approach | Cloning features | Scalability | Robust | Itinerary planning | Application |
|---|---|---|---|---|---|---|
| LCF [7] | Centralized | NO | Limited | Low | Static | Data fusion |
| GCF [7] | Centralized | NO | Limited | Low | Static | Data fusion |
| GA [8] | Centralized | NO | Limited | Low | Static | Data fusion |
| Xu et al. [15,16] | Distributed | NO | Limited | Good | Dynamic | Target tracking |
| MADD [17] | Distributed | NO | Limited | Good | Hybrid | Data fusion |
| Shakshuk et al. [18] | Distributed | NO | Limited | Good | Hybrid | Data fusion |
| NOID [9] | Centralized | NO | Good | Low | Static | Data fusion/aggregation |
| MIP [10] | Centralized | NO | Good | Low | Static | Data fusion/aggregation |
| BST-MIP [11] | Centralized | NO | Good | Low | Static | Data fusion/aggregation |
| DSG-MIP [12] | Centralized | NO | Good | Low | Static | Data fusion/aggregation |
| GA-MIP [13] | Centralized | NO | Good | Low | Static | Data fusion/aggregation |
| CBID [2] | Centralized | Yes | Good | Low | Static | Data fusion/aggregation |
| TBID [14] | Centralized | NO | Good | Low | Static | Data aggregation |
| IEMF [5] | Centralized | NO | Good | Low | Static | Data fusion |
| MMADIDD [19] | Distributed | NO | Good | Good | Dynamic | Data aggregation |

Gupta et al. [19] proposed a multiple mobile agents with dynamic itineraries-based data dissemination (MMADIDD) protocol where nodes are organized in a set of the wedge regions and each agent is responsible for gathering aggregated data from each wedge region. The route of an agent is dynamically decided at each hop using cost function. MMADIDD adapts to unexpected node failures during an agent migration, but consumes slightly more energy than TBID [14]. This protocol provides a fair amount of fault tolerance, but the migration of agents depends on the wedge structure.

The work described in this paper also uses a distributed approach where nodes are organized in a set of spanning trees using a distributed algorithm. In the case of node failures, each orphaned node invokes a local recovery mechanism to find the new parent node, in order to maintain the tree topology. For agent migration, our protocol uses a dynamic and distributed approach where the migration of agents is decided at each visited sensor node.

## 3. System Model and Assumptions

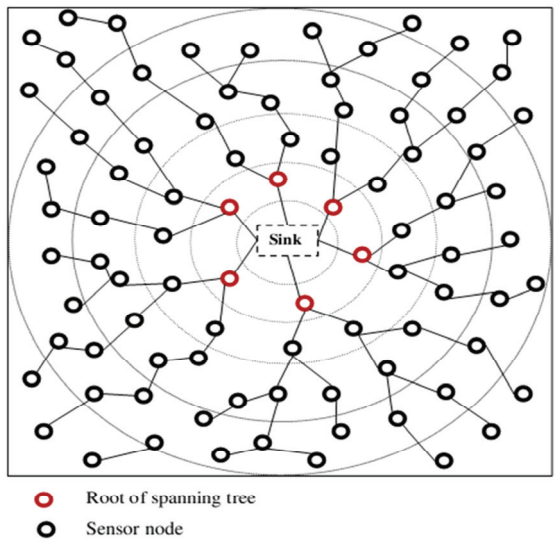



**Fig. 2.** Network topology.

**Table 2**. Data structures/variables used in processing

| Data structure/variable | Meaning |
|---|---|
| *senderNodeID* | Identifier for sender node |
| *nodeID* | Identifier for node |
| *levelNo* | Level number for the node |
| *parentID* | Identifier for the parent node |
| *rssi* | Received signal strength |
| *timeOn* | Flag the variable, which is initially false and that becomes true when the node broadcasts the tree creation packet. |
| *MA_direction* | Flag variable, which is false when an agent moves from the root to |

| | |
|---|---|
| | the leaf node, otherwise it is true. |
| $N_l$ | Number of 1-hop neighbor nodes of the sink |

We consider a sensor network that is composed of a set of sensor nodes and a processing element, called a sink node, which is placed at the center of the monitoring region. Each node is equipped with an omni-directional antenna. The sensor nodes, including the sink node in the network, have an equal communication range, such as $r_{max}$. The sink node has enough battery power and computational capability to handle all of the necessary calculations, while the sensor nodes in the network have limited battery power and computational capability. For our proposed scheme we used a network model similar to the model used in [2,14], and is shown in Fig. 2.

# 4. Proposed Protocol

In this section, we present a clone-based dynamic and distributed agent migration (CDDAM) scheme for periodic data aggregation and collection in WSNs. The operation of CDDAM consists of two phases: (1) the rooted spanning tree construction and (2) the clone- based agent migration phase. This section describes each phase in detail. The data structures used in processing are listed in Table 2.

## 4.1 Rooted Spanning Tree Construction

In this phase, the sink node creates an *RSTreeCreationPkt* packet, which contains the following three fields: *senderNodeID, levelNo*, and *parentID*. The field *senderNodeID* is the node ID of the sender node of this packet, *levelNo* is the number of hops to the root of the spanning tree, and *parentID* is the node ID of the sender's parent. The field *levelNo* and *parentID* is set to 0 by the sink node. The value of *levelNo* of any node indicates the level number of the node in the tree. After the creation of the *RSTreeCreationPkt* packet, the sink broadcasts it. Any node that is within the transmission range of the sink receives the *RSTreeCreationPkt* packet and then becomes the children of the sink node. Each of these nodes then broadcast an *RSTreeCreationPkt* packet, which advertises that they are at level one. The sink node hears these broadcasts and sets the senders as its children. All the nodes of level one become the root of their spanning tree. Nodes that have not joined the spanning tree yet, make the sender their parent after receiving these packets. If a node receives more than one packet, it selects the parent node whose packet was received with the highest received signal strength indicator (*rssi*) value. Equation 1 expresses this condition.

$$Parent(SN_{j,k}) = SN_{x,l} \mid (l < k \wedge \max\{x.rssi\}) \ \ for\ k,\ l \geq 1 \tag{1}$$

Here, $SN_{j,k}$ is sensor node $j$ at level $k$ of tree $T_i$ and $Parent(SN_{j,k})$ is the parent node of the sensor node $SN_{j,k}$ , $SN_{x,l}$ is node x at level $l$, and $l$ and $k$ are the level numbers of the sensor node. This process is repeated until all of the nodes have joined the spanning tree.

Each node maintains a *CandidatesForParent* table having the three fields of *nodeID*, *levelNo,* and *rssi*. This table keeps the information of only those neighbors whose *levelNo* is less than or equal to its own *levelNo*. The *rssi* is the received signal strength indicator of the received *RSTreeCreationPkt* packet. This

table is used to deal with the node failures. At the end of the tree formation phase, each node knows its parent node, its children, and its level number. Each node waits for time, $t_{delay}$ before broadcasting its *RSTreeCreationPkt* packet. The value of $t_{delay}$ depends on slot time assignment, $t_{st}$ of MAC protocol. The value of $t_{delay}$ is calculated as $t_{delay} = m^* t_{st}$. Here, $m$ is the maximum number of possible parent nodes in its transmission range. The pseudo-code for the rooted spanning tree construction phase is given in Algorithm 1.

**Algorithm 1: Rooted Spanning Tree Construction Process**

```
/* The following code is executed by the Sink node */
1: Create an RSTreeCreationPkt (sinkID,levelNo,parentID) packet
                    //for sink levelNo=0 and parentID=0
2:  Broadcast this packet to its 1-hop neighbors;
3:  If (receive RSTreeCreationPkt packet)
4:    add sender node ID in its children list;
5:      End If

/*The following code is executed by the sensor node (x) whenever an event (packet is received or timer expires) is detected */

Initially timeOn = false and levelNo = 0
1: If (receive RSTreeCreationPkt  packet  pkt)
2:   If (x.levelNo == 0)
3:      x.levelNo = pkt.levelNo +1; //Update level number
4:      x.parentID = pkt.senderNodeID;
5:      x.rssi = pkt.rssi;
6:      update CandidatesForParent table;
7:      set timer with time t_delay to wait before
                broadcasting this packet;
8:    End If
9:    If (x.levelNo +1 == pkt.levelNo)
10:     update CandidatesForParent table;
11:     If (x.rssi < pkt.rssi)
12:       x.parentID = pkt.senderNodeID;
13:       x.rssi = pkt.rssi;
14:     End If
15:   End If
16:   If (x.levelNo == pkt.levelNo)
17:    update CandidatesForParent table;
18:   End If
19:   If (x.levelNo < pkt.levelNo && x.nodeID == pkt.parentID)
20:    add sender node ID in its children list;
21:   End If
22:  End If
23: If (timeOn is false && timer t_delay expire)
24:    broadcast RSTreeCreationPkt(x.nodeID, x.levelNo,
            x.parentID) packet;
25:  timeOn = true;
26:  End If
```

After the construction of the rooted spanning trees, each node disseminates the depth of its sub-tree back towards the root of the tree. Leaf nodes initiate this by disseminating a depth of 1 to their parent. When a node has received a packet from each of its children reporting its depth, it decides on the maximum depth of its children and adds one for itself and sends its depth to its parent. Once the internal sensor node $SN_j$ of tree $T_i$ has received the depth of the sub-tree rooted at $SN_j$, it sets up the maximum expected waiting time ($t_{wait}$) taken by a slave agent to complete its migration within the sub-tree $T_{i_{subtree\ ,k}}$

## 4.2 Clone-Based Agent Migration Phase

In this phase, the sink node creates mobile agents, $MA_j$ for each rooted spanning tree, $T_i$. After that, the sink node dispatches an agent to each root of $T_i$ for data aggregation and collection tasks. An agent starts data aggregation and collection from the leaf node of the $T_i$.

Each $MA_j$ starts its migration from the root node of spanning tree $T_i$. When it reaches a sensor node $SN_i$ with two or more child nodes, it builds clones of itself $MA_{clone_{j,k}}$ similar to [2], where $k$ = 1, 2, ..$m$ and $m$ is number of children of the visiting node $SN_i$. Each clone agent $MA_{clone_{j,k}}$ is dispatched to visit the child node of the tree branch. However $MA_j$ remains on the parent node $SN_i$. When slave agent $MA_{clone_{j,k}}$ reaches a leaf node, it starts performing the data aggregation task and returns back to the parent node $SN_i$ where all of the data collected by $MA_{clone_{j,k}}$ is handed over to $MA_j$. After that the $MA_{clone_{j,k}}$ destroy it selves.

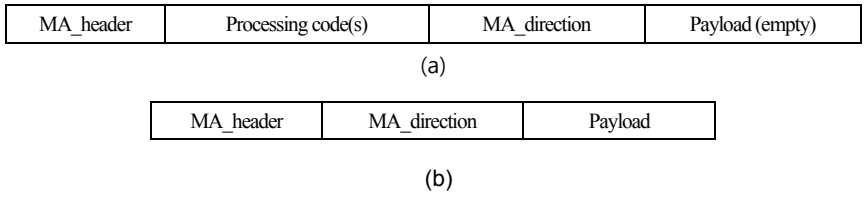



**Fig. 3.** Structure of a master mobile agent (MA) (a) when the MA_direction is DOWN, (b) when the MA_direction is UP.

The structure of mobile agent packet is shown in Fig. 3. An agent packet contains the following four fields: MA_header, Processing code, MA_direction, and Payload. Initially, the MA_direction is set to DOWN, which indicates the moving direction of an agent from the root to the leaf node of the spanning tree $T_i$. When an agent reaches the leaf node, it sets the MA_direction to UP and starts data aggregation from the leaf node towards the root node. During MA migration from the root to the leaf node, the processing code part of the agent packet is stored at each node. Therefore, there is no need to carry the processing code parts of an agent when an agent moves in the UP direction.

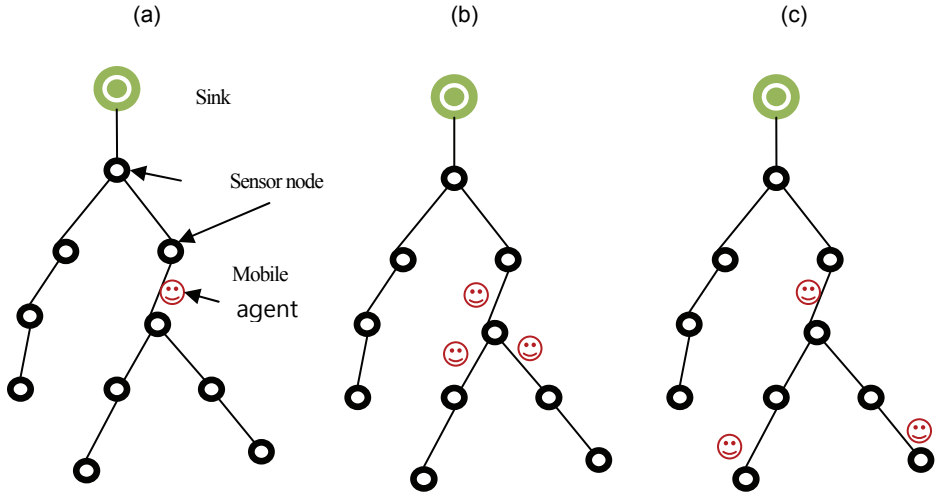
Algorithm 2 gives the details of the agent migration process within each tree $T_i$. Each $MA_j$ waits for its clone agent $MA_{clone_{j,k}}$ up to $t_{wait}$ maximum time, which can be computed by Eq. (2).

$$t_{wait} = 2 * depth_{T_i} \times (t_{trans} + t_{proc}) \quad (2)$$

Here, $depth_{T_i}$ is depth of sub-tree $T_i$, $t_{trans}$ is one hop transmission delay, and $t_{proc}$ is the time needed for the agent to complete its data aggregation at each node. Fig. 4 illustrates the agent migration in a tree, $T_i$.

## 4.3 Dealing with Node Failures

The sensor nodes are a resource-constrained device, which make them highly prone to sudden failures. The most common reason for node failure is due to energy depletion. Therefore, in the proposed scheme, whenever the residual energy of a node $SN_i$ goes down below a threshold energy level $E_{threshold}$, $SN_i$ broadcasts a beacon packet to inform its neighbors that it cannot be a part of the spanning tree $T_i$. After receiving this beacon packet from node $SN_i$, each node deletes the entry of the dying node from its *CandidatesForParent* table. When a node fails, its children become orphan nodes. For dealing with this case, each orphaned child ($u$) finds another sensor node ($v$) in its *CandidatesForParent* table with the maximum RSSI and *u.levelNo* > *v.levelNo* that should be connected to the spanning tree to act as a parent node. If u failed to find any parent node with above condition, it looks for a node v with maximum RSSI and u.levelNo = v.levelNo. But in case of later, there is some possibility that siblings may become parent of each other which may cause a loop. After finding the suitable parent node, each orphaned node $u$ sends an *AttachMe* packet to its new parent $v$. The *AttachMe* packet contains the following three fields: the orphaned node ID (*orphaned_nodeID*), the node ID of the new parent v (*newParentID*), and the depth of its sub-tree (*depth_of_tree*). After receiving the *AttachMe* packet from the orphaned node $u$, the parent node $v$ sends acknowledgement to $u$ and adds node $u$ in its list of children. If the new parent node $v$ finds any changes in its depth, it sends new depth to its parent and this process continues up to the root node of the spanning tree $T_i$.



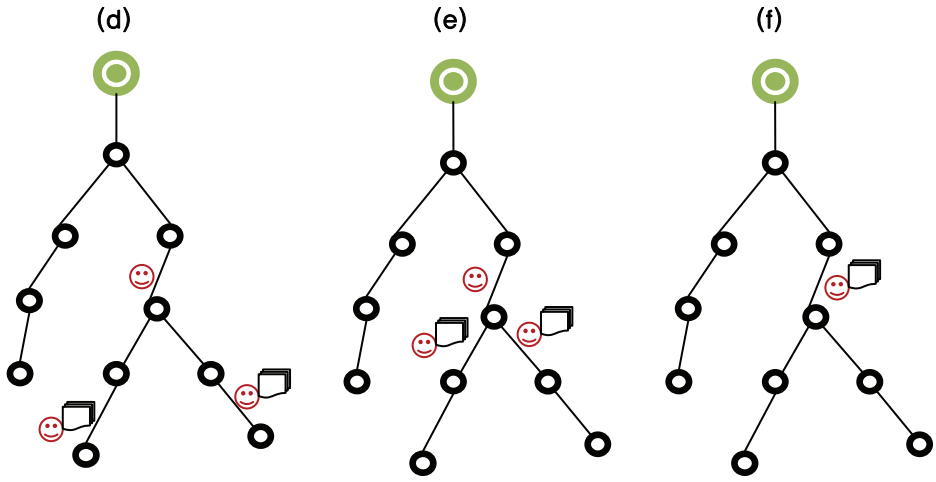



**Fig. 4.** Illustration of the clone-based agent migration scheme.

**Algorithm 2: Clone-Based Agent Migration Process**

**/*The following code is executed by the Sink node */**

Initialize: sink know the number of its 1-hop neighbors say $N_1$

1: k= 1;

2: **while** (k <= $N_1$) **do**

4: sink creates an agentPkt packet with maID = k and send to root node (i.e level one node) of spanning tree $T_k$;

5: k=k+1;

6: End while

**/*The following code is executed by the sensor node (x)*/**

1: **If** (node x receive *AgentPkt* packet p)

2: **If** (node is not leaf node and MA_direction is DOWN)

3: agent clone itself and dispatches slave Agents to visit each branches of tree.

4: **End If**

5: **If** (node is leaf node and MA_direction is DOWN)

6: set MA_direction to UP;

7: perform data aggregation;

8: send agent to its parent node;

9: **End If**

10: **If** (node is not leaf node and MA_direction is UP)

11: wait for all slave agents ($t_{wait}$);

12: perform data aggregation and combine the reduced data with data collected by slave agents;

13: send agent to its parent node;

14: **End If**

**15: End If**

# 5. Performance Analysis

This section presents the simulation results of our proposed protocol, CDDAM, and provides its comparison to CBID [2], TBID [15], and MMADIDD [19]. The analysis of the results is also provided here.

**Table 3.** Parameters for simulations

| Parameter | Value |
|---|---|
| Monitoring the field size | 200 × 200 |
| Number of nodes | [25, 200] |
| Node density | 0.0052 nodes/m2 |
| Transmission range | 25 m |
| Simulation time | 30 min |
| Initial battery power | 18720 J |
| Agent processing code, s (byte) | 1000 |
| Aggregated data accumulated by the agent at each node, d (byte) | [100, 200] |
| Data fusion coefficient (f) | 1 |
| Agent execution time at each sensor (ms) | 50 |
| MAC protocol | TMAC |

## 5.1 Simulation Environment

We have simulated and tested the proposed protocol using the Castalia simulator, v.3.2 [22]. The sensor nodes were randomly deployed over a square-monitoring field with a fixed node density (0.0052 nodes/$m^2$). The number of nodes was varied from 25 to 200 nodes to show the scalability property of the protocols. All of the sensor nodes had the same transmission range and battery power. For analyzing the results, each network scenario was executed 10 times and the results were plotted using a confidence interval of 95%. The simulation parameters used in our experiments are shown in Table 3.

We have used the following metrics to evaluate the performance of the protocols:

a) **Average energy consumption** ($E_{avg}$): this is measured as the amount of energy consumed by a node, due to communication, computation, and sensing in each round of data aggregation and collection. $E_{avg}$ is calculated using Eq. (3).

$$E_{avg} = (sum\ of\ energy\ consumption\ at\ each\ node)/(no\ of\ nodes) \tag{3}$$

b) **Overall response time** ($T_{overallresponsetime}$)**:** this is measured as the time taken to complete one round of data aggregation and collection tasks. $T_{overallresponsetime}$ is calculated using Eq. (4).

$$T_{overallresponsetime} = \left(\sum_{i=1}^{n} \max_{j=1\ldots k}\{T_{i,j}\}\right) \Big/ n \tag{4}$$

Here, $T_{i,j}$ is time taken by the $j^{th}$ agent of the $i^{th}$ round to complete its data aggregation task and $n$ is the number of rounds.

c) **Success rate of agent's round trip** ($S_{rate}$)**:** this is measured as the percentage of agents that were received by the sink over the total number of agents dispatched.

$$S_{rate} = (N_{receivedAgents} / N_{dispatchedAgents}) * 100 \quad (5)$$

Here, $N_{receivedAg\ ents}$ is the number of agents successfully received at the sink after completing their round $N_{dispatchedAgents}$ trip and is the total number of agents dispatched by the sink in the network.

## 5.2 Impact of the Network Size

Fig. 5(a) and (b) shows the performance comparison of CDDAM with CBID, TBID, and MMADIDD, in terms of the average energy consumption with the varying number of nodes for different accumulated data, d (i.e. for d=100 and d=200, respectively). It demonstrates that average energy consumption increases as the number of node increases. This occurs due to the increase in the length of an agent's itinerary and the number of nodes visited by an agent. The CDDAM protocol takes approximately 8% less energy than CBID, as shown in Fig. 5(a). This is due to the use of a shorter agent packet in the CDDAM protocol, which results in lower energy consumption than CBID. However, CBID is a centralized approach-based protocol where an agent needs to carry a pre-computed itinerary list, as well as cloning information, which increases the size of the agent packet. This results in extra transmission energy consumption for the agent migration within the network. Moreover, we observed that CDDAM consumes approximately 14% less energy than TBID. This is because CDDAM dynamically computes the routes of an agent at each host and uses the agent-cloning concept. Also, the agent's itinerary length is shorter than TBID and MMADIDD. Due to the use of the cloning concept, each clone agent (SA) needs to carry a lighter payload. Furthermore, CDDAM consumes approximately

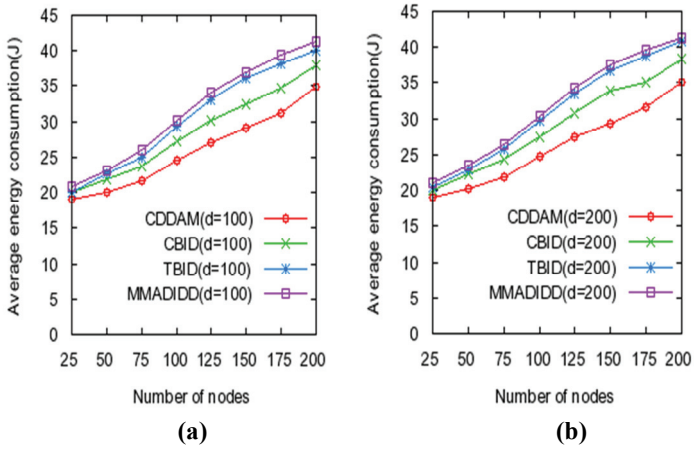



**Fig. 5.** Performance of CDDAM and TBID in terms of average energy consumption for (a) d=100 and (b) d=200. CDDAM=clone-based dynamic and distributed agent migration, CBID=clone- based itinerary design, TBID=tree-based itinerary design, MMADIDD=multiple mobile agents with dynamic

itineraries-based data dissemination.

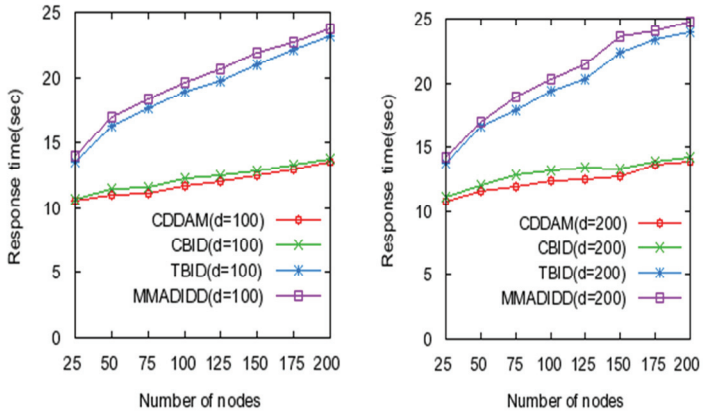



**Fig. 6.** Performance of CDDAM and TBID in terms of overall response time for (a) d=100 and (b) d=200. CDDAM=clone-based dynamic and distributed agent migration, CBID=clone-based itinerary design, TBID=tree-based itinerary design, MMADIDD=multiple mobile agents with dynamic itineraries-based data dissemination.

16% less energy than MMADIDD. This is because MMADIDD does not use the agent-cloning concept and agent's itinerary length is longer, as compared to CDDAM.

Fig. 6(a) and (b) shows the performance comparison of CDDAM with CBID, TBID, and MMADIDD in terms of the overall response time for d=100 and d=200, respectively. It demonstrates that the overall response time increases as the number of nodes increases. This is due to the increase in the number of nodes visited by an agent. The overall response time of CDDAM is approximately 2% less than CBID. This is because an agent does not need to carry a pre-computed itinerary list and cloning information in the CDDAM protocol. However, CDDAM takes approximately 36% less time than TBID to complete one round of data aggregation and collection tasks. Moreover, the overall response time of MMADIDD is approximately 38% more than CDDAM. This is because CDDAM uses the agent-cloning concept, which offers a parallel deployment of the clone agent to complete the task, which results in a better response time than other schemes.

## 5.3 Impact of Faulty Nodes

Fig. 7 shows the performance comparison of CDDAM with CBID, TBID, and MMADIDD in terms of the success rate of the agents' trip in the presence of faulty nodes. It can be observed from Fig. 6 that CDDAM and MMADIDD perform better than the protocols for other schemes in the presence of faulty nodes. This is because CDDAM and MMADIDD protocols are distributed algorithm based protocols where next the hop to be visited by an agent is dynamically decided at each host. However, CBID and TBID protocols are centralized algorithm based protocols where agents follow pre-computed routes for the migration of agents, which reduces the success rate in the presence of faulty nodes.

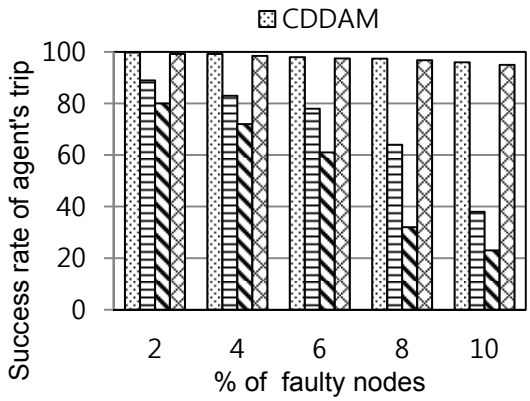



**Fig. 7.** Success rate of an agent's trip in the presence of faulty nodes. CDDAM=clone-based dynamic and distributed agent migration, CBID=clone-based itinerary design, TBID=tree-based itinerary design, MMADIDD=multiple mobile agents with dynamic itineraries-based data dissemination.

## 6. Conclusion

This paper presents a dynamic and distributed algorithm-based agent migration scheme, CDDAM for data aggregation in WSN. This scheme creates a virtual infrastructure, called rooted spanning trees, $T_i$, to perform the migration of agents within the network. CDDAM is a distributed version of the existing centralized-approach based migration scheme of CBID [2]. CDDAM not only provides a fault tolerance, but also significantly improves the performance when compared to CBID in terms of the average energy consumption, overall response time, and success rate of the agents' trips. The local information of the node is used to deal with the faulty nodes in this scheme. We have also studied the behavior of the proposed scheme through extensive simulation experiments and compared the performance with the three well-known schemes of CBID, TBID, and MMADIDD. The simulation results demonstrate that CDDAM has superior performance than the other schemes in the presence of faulty nodes. In our future work, we plan to incorporate security features in CDDAM to thwart the agents' migration to malicious host nodes.

## References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393-422, 2002.

[2] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "CBID: a scalable method for distributed data aggregation in WSNs," *International Journal of Distributed Sensor Networks*, vol. 2010, 2010.

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

[4] Y. Xu and H. Qi, "Distributed computing paradigms for collaborative signal and information processing in sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64, no. 8, pp. 945-959, 2004.

[5] M. Chen, L. Yang, T. Kwon, L. Zhou, and M. Jo, "Itinerary planning for energy-efficient agent communication in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 7, pp. 3290-3299, 2011.

[6] H. Qi, S. Iyengar, and K. Chakrabarty, "Multiresolution data integration using mobile agents in distributed sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 31, no. 3, pp. 383-391, 2001.

[7] H. Qi and F. Wang, "Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks," in *Proceedings of the 13th International Conference on Wireless Communications*, Calgary, Canada, 2001, pp. 147-153.

[8] Q. Wu, N. S. Rao, J. Barhen, S. S. Iyenger, V. K. Vaishnavi, H. Qi, and K. Chakrabarty, "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.

[9] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "Deriving efficient mobile agent routes in wireless sensor networks with NOID algorithm," in *Proceedings of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC2007)*, Athens, Greece, 2007, pp. 1-5.

[10] M. Chen, S. Gonzalez, Y. Zhang, and V. C. Leung, "Multi-agent itinerary planning for wireless sensor networks," in *Proceedings of International Conference on Quality of Service in Heterogeneous Networks (QSHINE2009)*, Las Palmas, Spain, 2009, pp. 584-597.

[11] M. Chen, W. Cai, S. Gonzalez, and V. C. Leung, "Balanced itinerary planning for multiple mobile agents in wireless sensor networks," in *Ad Hoc Networks*, Heidelberg: Springer, 2010, pp. 416-428.

[12] M. Chen, S. Gonzalez-Valenzuela, and V. C. Leung, "Directional source grouping for multi-agent itinerary planning in wireless sensor networks," in *Proceedings of IEEE International Conference on Information and Communication Technology Convergence (ICTC2010)*, Jeju, Korea, 2010, pp. 207-212.

[13] W. Cai, M. Chen, T. Hara, and L. Shu, "GA-MIP: genetic algorithm based multiple mobile agents itinerary planning in wireless sensor networks," in *Proceedings of the 5th Annual International Wireless Internet Conference (WICON2010)*, Singapore, 2010, pp. 1-8.

[14] C. Konstantopoulos, A. Mpitziopoulos, D. Gavalas, and G. Pantziou, "Effective determination of mobile agent itineraries for data aggregation on sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 12, pp. 1679-1693, 2010.

[15] Y. Xu and H. Qi, "Dynamic mobile agent migration in Wireless Sensor Networks," *International Journal of Ad Hoc Ubiquitous Computing*, vol. 2, no. 1, pp. 73-82, 2007.

[16] Y. Xu and H. Qi, "Mobile agent migration modeling and design for target tracking in wireless sensor networks," *Ad Hoc Networks*, vol. 6, no. 1, pp. 1-16, 2008.

[17] M. Chen, T. Kwon, Y. Yuan, Y. Choi, and V. C. Leung, "Mobile agent-based directed diffusion in wireless sensor networks," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, pp. 219-219, 2007.

[18] E. Shakshuki, H. Malik, and M. K. Denko, "Software agent-based directed diffusion in wireless sensor network," *Telecommunication Systems*, vol. 38, no. 3-4, pp. 161-174, 2008.

[19] G. Gupta, M. Misra, and K. Garg, "Multiple mobile agents based data dissemination protocol for wireless sensor networks," in *Advances in Computer Science and Information Technology: Networks and Communications*, Heidelberg: Springer, 2012, pp. 334-345.

[20] X. Wang, M. Chen, T. Kwon, and H. Chao, "Multiple mobile agents' itinerary planning in wireless sensor networks: survey and evaluation," *IET Communications*, vol. 5, no. 12, pp. 1769-1776, 2011.

[21] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, Boston, MA, 2000, pp. 56-67.

[22] A. Boulis, "Castalia 3.2: a simulator for wireless sensor networks and body area networks," Mar. 2011; https://castalia.forge.nicta.com.au/index.php/en/news/77-castalia-3-2-released-30-3-2011.html.

**Govind Gupta** http://orcid.org/0000-0002-0456-1572

He is currently a Ph.D. candidate in the Department of Computer Science & Engineering at Indian Institute of Technology, Roorkee, India. His research interests include mobile computing, wireless ad hoc and sensor network, network security, distributed computing and performance evaluation. He is a student member of IEEE.

**Manoj Misra**

He received his Bachelor's degree in Electrical Engineering in 1983 from HBTI Kanpur, India and Master's in Computer Science in 1986 from University of Roorkee, India. He received his Ph.D. in Computer Science in 1997 from University of Newcastle, Upon Tyne, UK. He is currently a Professor in the Department of Computer Science and Engineering at Indian Institute of Technology, Roorkee. He has guided several PhD theses, ME/MTech dissertations. His areas of interest include mobile computing, distributed computing, wireless ad hoc and sensor network and performance evaluation. He is a senior member of IEEE.

**Kumkum Garg**

She received her Bachelor's in Electronics Engineering from University of Roorkee, India in 1971 and Master's in Computer Engineering in 1977 from University of Roorkee, India. She received her Ph.D. in Computer Engineering in 1984 from University of London, UK. She has been a Professor in the Department of Computer Science and Engineering, Indian Institute of Technology, and Roorkee, India from 1989 to 2010. She has been the Head of Information superhighway Centre from 2005 to 2006 at IIT, Roorkee. Currently, she is Professor of Computing and Dean, Faculty of Engineering at Manipal University Jaipur, India. She has guided several PhD theses, ME/MTech dissertations and completed various projects. Her research interests include mobile computing, mobile agents, network security, trust, wireless ad hoc and sensor network. She is a senior member of the IEEE Computer Society.