Efficient Implementation of Data Aggregation in WSNs by Mobile Agent Paradigm

N.Sandhya rani
Dept. of CSE, UCEV - JNTUK
Vizianagaram, A.P., India
sandhya.nelli@gmail.com

O. Srinivasa Rao
Dept. of CSE, UCEV - JNTUK
Vizianagaram, A.P., India
osr_phd@yahoo.com

Dr MHM Krishna Prasad
Dept. of IT, UCEV - JNTUK
Vizianagaram, A.P., India
krishnaprasad.mhm@gmail.com

***Abstract*— In many sensor network applications viz., environmental monitoring, spatial exploration and battlefield surveillance, sensed data is aggregated and transmitted to the sinks for analysis. Thus, in-network data aggregation [2] becomes an important technique in wireless sensor networks and has been well studied in recent years. In general, any sensory network suffers with two problems i.e., i) the time latencies of the existing scheduling algorithms are still high, and ii) the other one is that all the existing algorithms are centralized, which are inherently inefficient. Hence, in this paper, we present a distributed scheduling algorithm generating collision-free schedules with latency bound, based on TBID, which employs multiple Mobile Agents (MAs) for completing the aggregation task on WSNs. The algorithm not only determines the proper number of MAs for minimizing the total aggregation cost but also constructs low-cost itineraries for each of them. In this paper, we adopt TBID that improves upon NOID by following a more direct approach to the problem of determining low-cost MA itineraries. Specifically, based on an accurate formula for the total energy expended during MA migration, we follow a greedy-like approach for distributing SNs in multiple MA itineraries, and algorithm determines a spanning forest of trees in the network, calculates efficient tree traversal orders (itineraries), and eventually, assigns these itineraries to individual MAs.*)**

Keywords-component; wireless sensor networks, data aggregation, mobile agents

I. INTRODUCTION

Wireless Sensor Networks (WSNs) have prominence role in networking paradigm. A new concept collaborative processing introduced with WSNs. The collaborative processing performs the correlation inherent within the information of nearby sensors because their sensing regions are largely overlapping. Data from neighboring Sensor Nodes (SNs) refer to the same source of information. So, an aggregation is performed on the original sensor data. By applying aggregation on the original data gradually reduces the communicated data and relieves the network from the large amount of data. Then this data directly send to the processing element through mobile agents. Recently, the use of Mobile Agents (MAs) [1] has been suggested as an intrinsically distributed computing technology in the field of WSNs [3]. An MA [1] is an autonomous program moving from node to node and collects data from these sensor nodes. The Mobile Agent performs the local processing in each SN to get aggregated data. Then this data carried by the MA to the next SN, where the MA program performs aggregation upon the new data. By changing the program code of MAs, different processing can be carried out at each SN. Otherwise, SNs should have stored the program code of each possible application, resulting in increased memory demands in each SN. Due to this ease of reprogramming, the MA paradigm has been incorporated in several middleware architectures for WSNs. When employing MAs for aggregation operations on SNs, the choice of agents' itineraries is of critical importance affecting the overall energy consumption and aggregation cost. In this paper, we propose a heuristic algorithm that employs multiple MAs for completing the aggregation task on WSNs. Our algorithm, the Tree-Based Itinerary Design (TBID) algorithm, not only determines the proper number of MAs for minimizing the total aggregation cost but also constructs low-cost itineraries for each of them. It also improves on a previous algorithm [7], [8] of ours, where a completely different approach is presented for the same objective of low-cost MA itineraries. With the use of right data structures, the computational complexity of our algorithm can be kept low, and thus, the adopted algorithm is suitable for WSNs with dynamic network conditions. The high effectiveness of our algorithm not only determines the proper number of MAs for minimizing the total aggregation task but also constructs low cost itineraries for each of them.

II. ALGORITHM

In this project, we adopted the TBID algorithm. The main idea of TBID is to partition the area around the Processing Element (PE) into concentric zones (Fig. 1.) and start building the MA paths to be directed from the inner (close to PE) zones to the outer ones. We adopted algorithms [6] for Initialization, phase 1, phase 2 and Processing cost updation. In Algorithm 1, we formally define the concentric zones Z_i, *(i= 1 . . .L)* around the PE, where L is the total number of zones. The radius of the first zone that includes the PE is equal to r_{max}, where is an input parameter in the range *(0,1]* and r_{max} is the maximum transmission range of any SN. The thickness of all other zones is $r_{max}=2$. So, each SN in a zone Z_i can only communicate with nodes belonging to the previous, current, and the next outer zones, namely zones Z_{i-1}, Z_i, Z_{i+1}.

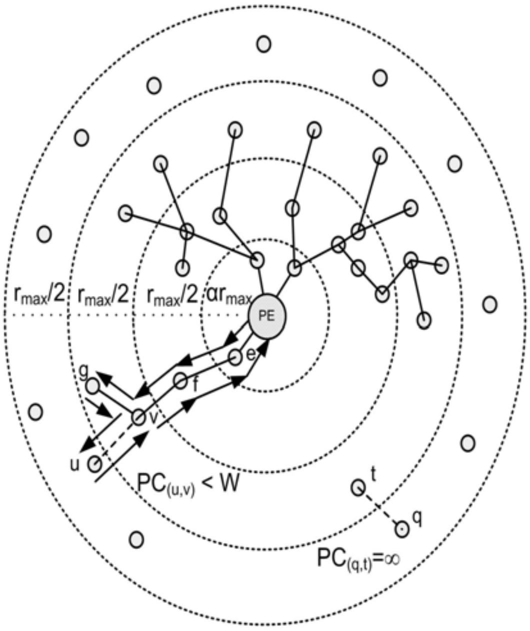


Figure 1. Partitioning the area around PE into a number of concentric zones

The main processing consists of two phases. The first phase (Algorithm 2) starts from the inner zone Z_1 and proceeds to the outer zones Z_i, (i=2. . . L). The algorithm of phase 1 is executed on each zone. The end result of phase 1 in the first zone is to connect all SNs in this region directly to the PE. The SNs in the first zone will be the starting points of the itineraries that MAs will follow. Thus, TBID will create as many MAs as the number k of these nodes (k =3 in Fig. 1.).

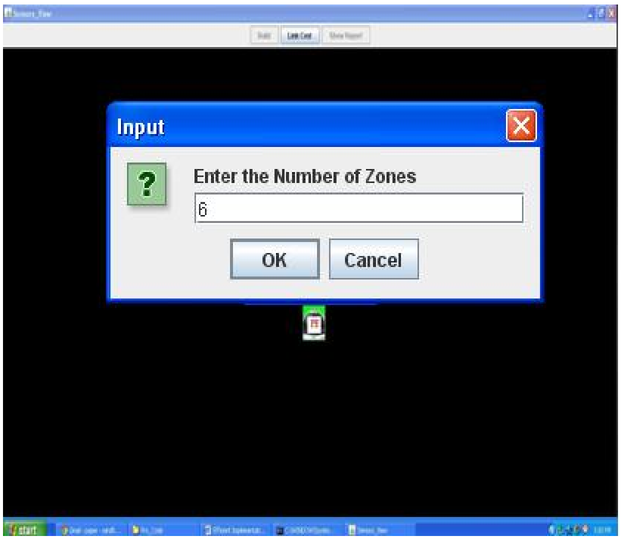
III. IMPLEMENTATION DETAILS

TBID effectively executes N rounds, at each round an SN is attached to a tree. The node attachment can be done in binary tree scheme. At each round of phases 1 and 2[6], one of the trees grows with the attachment of an SN to that tree. Each tree corresponds to an MA. The MA path that starts with PE and continues with the post order traversal of the tree with shortcutting, whenever possible, determines the itinerary of the MA. At each round, first, the lowest PC value should be determined. After attaching an SN to an IT, the PC values should be updated for all edges *(u ,v)* connecting an SN u not yet attached to any IT with an SN v belonging to the just expanded tree. Then the SN with the lowest PC value should be attached to a tree. Finally, all PC values of the affected edges should be updated. Another important operation executed at each iteration of TBID is the estimation of the minimum *PC(u ,v)* value. This minimum determines the SN that will be next attached to an IT. As this operation is executed repeatedly, a heap structure is used for storing PC values so that extracting the minimum

PC value can be executed fast in *O(log t)* time, where *t* is the number of values stored in the heap. Then, we describe data structures for identifying fast the minimum PC value among all edges of the SN graph.

IV. EXPERIMENTAL WORK

The TBID Algorithm is implemented using Java (jdk 1.5) and all the experiments are performed on PC with Pentium IV CPU and 512MB RAM running on the operating system Windows XP. The experimental results are shown in below figures.



Input

Enter the Number of Zones

6

Figure 2.1. PE activated with 6 zones

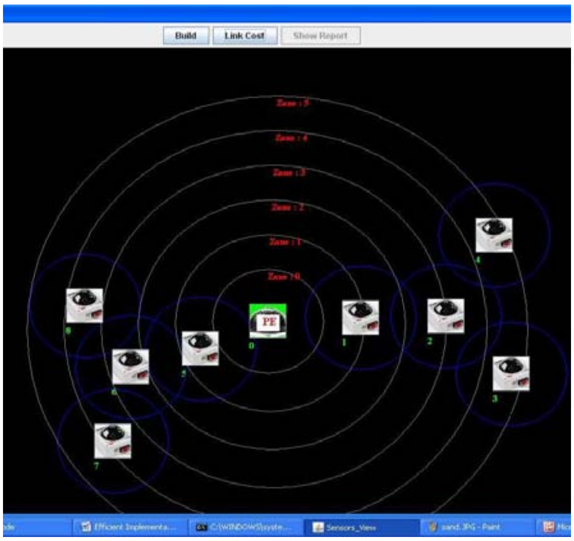


Figure 2.2. PE with the sensors around it

The above Fig. 2.1 shows that first we have to place processing element.Then we have to enter number of zones.6 zones are entered in the text box.

The above Fig. 2.2 shows that we are placing sensors in the zones.

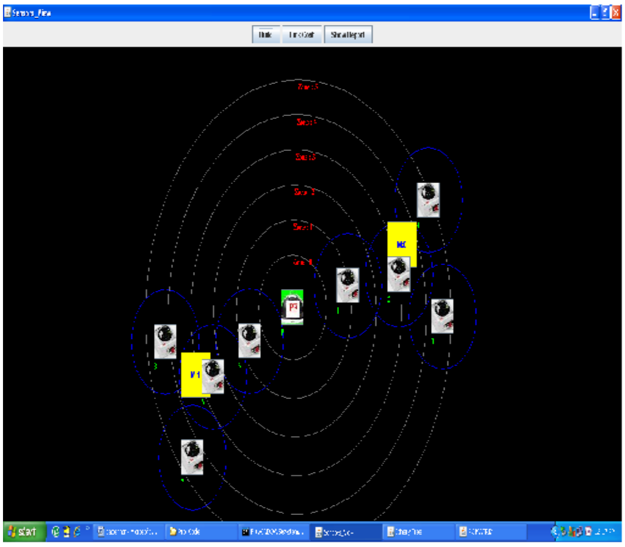


Figure 2.3. Mobile Agents moving along the tree path to collect the data from the sensors

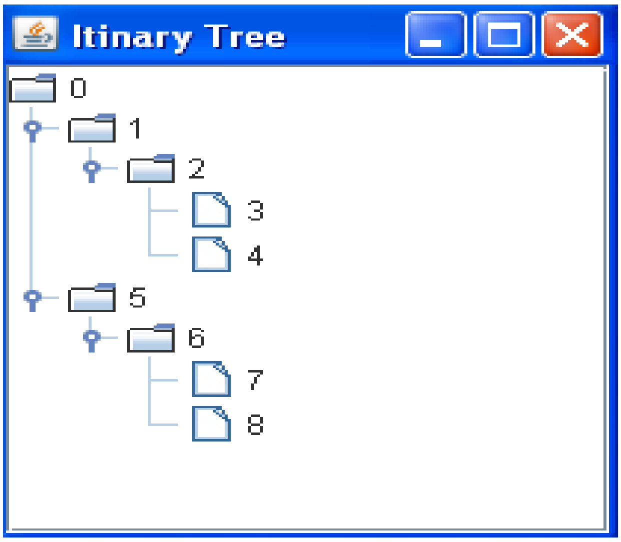
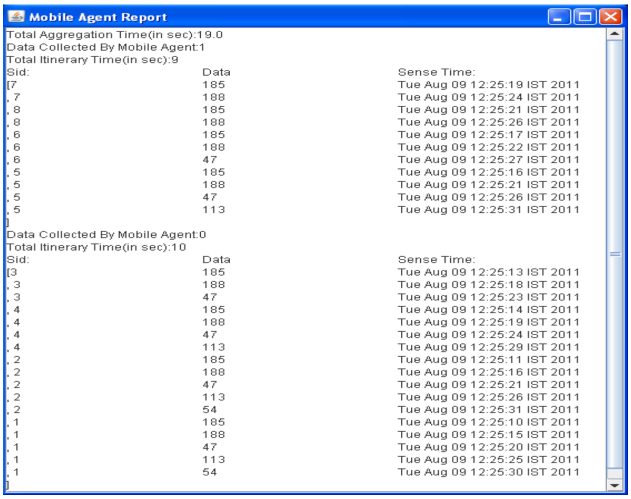


Figure 2.3. The Itinerary Tree formed is displayed

The Fig. 2.3 shows that two sensors are there in zone 0.So two mobile agents are migrated among the sensors and collect the data from those sensors and performing data aggregation at each sensor.

The Fig. 2.4 shows that an itinerary tree is formed by connecting those sensors.



```
Total Aggregation Time(in sec):19.0
Data Collected By Mobile Agent:1
Total Itinerary Time(in sec):9
Sid:                    Data                    Sense Time:
[7                      185                     Tue Aug 09 12:25:19 IST 2011
, 7                     188                     Tue Aug 09 12:25:24 IST 2011
, 8                     185                     Tue Aug 09 12:25:21 IST 2011
, 8                     188                     Tue Aug 09 12:25:26 IST 2011
, 6                     185                     Tue Aug 09 12:25:17 IST 2011
, 6                     188                     Tue Aug 09 12:25:22 IST 2011
, 6                     47                      Tue Aug 09 12:25:27 IST 2011
, 5                     185                     Tue Aug 09 12:25:16 IST 2011
, 5                     188                     Tue Aug 09 12:25:21 IST 2011
, 5                     47                      Tue Aug 09 12:25:26 IST 2011
, 5                     113                     Tue Aug 09 12:25:31 IST 2011
]
Data Collected By Mobile Agent:0
Total Itinerary Time(in sec):10
Sid:                    Data                    Sense Time:
[3                      185                     Tue Aug 09 12:25:13 IST 2011
, 3                     188                     Tue Aug 09 12:25:18 IST 2011
, 3                     47                      Tue Aug 09 12:25:23 IST 2011
, 4                     185                     Tue Aug 09 12:25:14 IST 2011
, 4                     188                     Tue Aug 09 12:25:19 IST 2011
, 4                     47                      Tue Aug 09 12:25:24 IST 2011
, 4                     113                     Tue Aug 09 12:25:29 IST 2011
, 2                     185                     Tue Aug 09 12:25:11 IST 2011
, 2                     188                     Tue Aug 09 12:25:16 IST 2011
, 2                     47                      Tue Aug 09 12:25:21 IST 2011
, 2                     113                     Tue Aug 09 12:25:26 IST 2011
, 2                     54                      Tue Aug 09 12:25:31 IST 2011
, 1                     185                     Tue Aug 09 12:25:10 IST 2011
, 1                     188                     Tue Aug 09 12:25:15 IST 2011
, 1                     47                      Tue Aug 09 12:25:20 IST 2011
, 1                     113                     Tue Aug 09 12:25:25 IST 2011
, 1                     54                      Tue Aug 09 12:25:30 IST 2011
]
```

Figure 2.4. Execution Time for computation

The Fig. 2.4 shows the each sensor data and the total execution time for total computation of mobile agents.

A. Execution Time:

Our experimental results, which are obtained from the TBID Algorithm is shown in the Fig.-3(a) and 3(b).

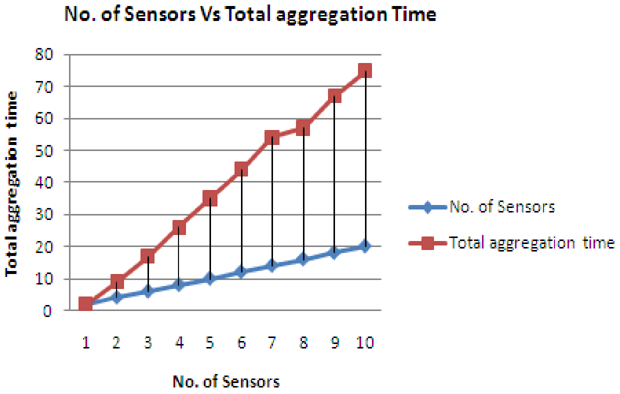


Figure 2. Aggregation time Vs number of sensors

Fig. 3, shows the Aggregation time increases as the number of sensors increased with two mobile agents and number of zones=10.

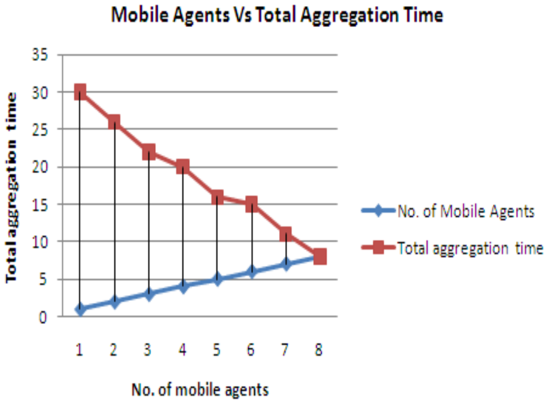


Figure 3. Aggregation time Vs number of mobile agents

From Fig. 4, one can observe that the Aggregation time decreases as the number of mobile agents increased, e.g., with no. of sensors=8 and no. of zones=10

V. CONCLUSION

We have presented a distributed scheduling algorithm generating collision-free schedules with latency bound, based on TBID by employing multiple Mobile Agents (MAs) for completing the aggregation task on WSNs. The algorithm not only determines the proper number of MAs for minimizing the total aggregation cost but also constructs low-cost itineraries for each of them. Simulation results confirm the high effectiveness of the adopted algorithm over other alternative solutions proposed in the literature. Also, with the use of proper data structures, the computational complexity of the algorithm is kept low.

VI. REFERENCES

[1] V. Pham, and A. Karmouch, “Mobile Software Agents: An Overview,” IEEE Comm. Magazine, vol. 36, no. 7, pp. 26-37, July1998.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] R. Rajagopalan and P. Varshney, “Data-Aggregation Techniques in Sensor Networks: A Survey,” IEEE Comm. Survey & Tutorials, vol. 8, no. 4, pp. 48-63, Oct. 2006

[3] H.Qi,Y. Xu, and X. Wang, “Mobile-Agent-Based Collaborative Signal and Information Processing in Sensor Networks,” Proc. IEEE, vol. 91, no. 8, pp. 1172-1183, Aug. 2003.

[4] 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, Jan. 2008.

[5] K. Kalpakis, K. Dasgupta, and P. Namjoshi, “Efficient Algorithms for Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks,” Computer Networks, vol. 42, no. 6, pp. 697-716, Aug. 2003.

[6] Charalampos Konstantopoulos, Aristides Mpitziopoulos, Damianos Gavalas, Member, IEEE, and Grammati Pantziou “Effective Determination of Mobile Agent Itineraries for Data Aggregation on Sensor Networks” VOL. 22, NO. 12, DECEMBER 2010

[7] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G.Pantziou, “Deriving Efficient Mobile Agent Routes in Wireless Sensor Networks with NOID Algorithm,” Proc. IEEE 18th Ann. Int’l Symp. Personal, Indoor, and Mobile Radio Comm. (PIMRC ’07), pp. 1-5, Sept. 2007.

[8] D. Gavalas, G. Pantziou, C. Konstantopoulos, and B. Mamalis, “A Method for Incremental Data Fusion in Distributed Sensor Networks,” Proc. IFIP Third Conf. Artificial Intelligence Applications and Innovations (AIAI ’06), pp. 635-642, June 2006.