Mobile Sink Scheduling Method for Wireless Sensor Networks under Travel Time Uncertainty

Murat Karakaya

***Abstract*—In Wireless Sensor Networks (WSN), timely data collection is an important requirement for the success of the applications. On the other hand, sensor nodes have limited resources, such as battery power and memory capacity which limits their direct participation to data collection process. Therefore, to collect sensory data efficiently, well-designed techniques are to be developed. One of the proposed techniques is to employ mobile sinks (MS) to decrease the energy consumption at sensor nodes spent in forwarding data packages. In this method, MS is scheduled such that it arrive sensor nodes before their memory gets full and overflows. For a successful schedule, the crucial information is the travelling time between sensor nodes in the field. In most cases, it is assumed that the travelling time is known a priori and remains the same all the time. However, in reality, due to various reasons, travelling times can change in course of time and, hence, the planned schedule may not produce the desired output. In this study, we propose an improved scheduling method considering uncertainty in travelling time. Simulation experiments justify the expected success of the proposed method.**

***Index Terms*—Wireless sensor networks, mobile sinks, scheduling, data collection, uncertainty.**

I. Introduction

Sensor nodes (SN) in a Wireless Sensor Network (WSN) collect data from their environment and store these values in their memory. If WSN employs a mobile sink (MS) to collect sensory data from SNs, MS visits SNs, collects sensory data, resets SN’s memory, and transfers the data to a remote center. Contrary to using a static sink (SS), employing a MS saves SN’s battery which could have been used in transferring sensory data to a SS via multi-hop communications.

In order to collect sensory data, MS can be routed according to a schedule. The schedule should provide timely data collection since memory capacity of SN is limited. If MS is scheduled to visit a SN too late, memory of the SN gets overflow and it is reset. Thus, all the collected data is erased and lost. To prevent memory overflows or minimize the number of overflows, MS must be scheduled carefully.

Somasundara *et al*. propose several heuristics to route MS by selecting the next SN to visit [1], [2]. In their work, the next SN is decided according to two important factors: SN distance from the current MS location and remaining time to overflow criteria. Their proposed heuristics do not produce a schedule. Moreover, MS must know remaining overflow times of all SNs which require continuous information exchanges with SNs. As a result of this requirement, energy consumption of these heuristics would be very high which makes these heuristics unpractical. To remedy this, we develop a novel heuristic, called MSCT, which creates a schedule for visiting SN such that the SN included in the schedule can periodically be visited in a time less than the time a SN memory gets full [3]. Thus, SN in the schedule is free of memory overflow. In [3], it is showed that MSCT can produce better results compared to proposed heuristics in [1] and [2].

In above mentioned studies, it is assumed that the travelling time between SNs are known a priori and they remain fixed all the times. However, in reality, MS can encounter various problems while travelling in the field which can delay it. For instance, if sensor nodes are deployed in a remote area, roads can be blocked by fallen trees or rocks which cause MS deviate from the planned route and get delayed. Therefore, uncertainty in travelling times should be considered while planning the MS route and while executing the schedule.

In this work, MSCT method is improved considering uncertainty in the travelling time as an important design parameter. The details of the proposed method, MSCT with Travelling Time Uncertainty (MSCT/U), are presented in Section II. The simulation setup and results of the experiments are provided in Section III followed by the conclusions.

II. MSCT with Travelling Time Uncertainty (MSCT/U)

A. Maximum Sensor Coverage Tour (MSCT)

As explained above, MSCT attempts to create a closed tour which takes less time than the overflow time of a sensor node [3]. The created tour is repeated by MS continuously. Thus, all the SNs in this tour are guaranteed to be re-visited before any of them experiences a memory overflow. To implement this idea efficiently, MSCT uses the Nearest Neighbor (NN) heuristic to select the next SN. Instead of using location of sensor nodes dispersed in a large field, MSCT calculates the nearest SN according to its beacon signal power. Thus, MSCT does not need to know all the location information of the deployed SNs. Moreover, since MSCT constructs a tour and repetitively executes it, MSCT does not require SN to broadcast their remaining time to overflow. Thus, MSCT saves SN’s battery power removing all communication requirements that is essential for the previously proposed methods in [1] and [2]. MSCT counts on the success of the NN heuristic to create a tour which includes maximum number of SNs. However, NN assumes that the travelling times between SNs are fixed. As discussed above, in certain situations, this could be contrary to the facts. Below, how MSCT is improved to deal with travelling uncertainty is

Manuscript received June 9, 2015; revised December 12, 2015.

M. Karakaya is with the Department of Computer Engineering, Atilim University, Ankara, Turkey (e-mail: murat.karakata@atilim.edu.tr).

explained.

B. Improving MSCT

MSCT is improved in two phases: while creating the initial tour, and while repeating the constructed tour.

Creating the initial tour phase: MSCT first calculates the maximum tour time (MTT) using sensor node's sensing rate (SR) and memory capacity (MC) as follows:

$$\mathrm{MTT} = \frac{\mathrm{MC}}{\mathrm{SR}} \tag{1}$$

NN selects nearest node to add the initial tour until the total travelling times allows returning to the start node. Thus, initial tour begins from a start node, visits sensor nodes, and returns to the start node. The total tour time (TTT) must be less than MTT to prevent any overflow happen at the SNs included in the tour. However, NN selects a SN according to expected travel time (ETT). If there is a delay on the road this assumption fails. If the delay happens during the creation of the initial tour, TTT may be larger than MTT which means that all the SNs included in the tour will overflow. To prevent this, when the initial tour is finalized, MSCT/U checks TTT. If it is larger than MTT, MSCT/U prunes the last node in the tour and recalculates the TTT. If, still, TTT is larger, MSCT/U continues to prune the tour until the expected TTT is less than MTT. Thus, in the end, we have an initial tour whose TTT is less MTT, which maintains the expected function of original MSCT under travelling time uncertainty.

Repeating the constructed tour: In real life, delay may occur at any time. Therefore, after constructing the initial route, MSCT/U improves the route as follows. First, if MS experienced a delay between two consecutive SNs in the route in the last tour, MSCT/U assumes that the delay can last for the next tour as well. Therefore, to refrain from the delay, this part of the route is updated as seen in Fig. 1. If there was a delay between nodes A and B, MSCT/U searches two unvisited SNs such that the total expected arrival time (EAT) from A to C via X and Y will be smaller than the realized arrival time (RAT) from A to C via B as in Eq. 2.

$$\alpha \times \mathrm{EAT}_{\mathrm{AXYC}} \leq \mathrm{RAT}_{\mathrm{ABC}} \tag{2}$$

where α is a coefficient and $\alpha \geq 1$ to tolerate possible delays that the new route (A-X-Y-Z) may introduce. If MSCT/U finds such a node, it replaces the node B with the nodes X and Y. As a result, the updated tour has one more node to cover which decreases number of overflows and increases the amount of collected data.

If MSCT/U does not find such nodes, it attempts to minimize RTT by swapping B with another node such that updated tour will take less time as in Fig. 2. In this case, X must satisfy the condition given Eq. 3.

$$\alpha \times \mathrm{EAT}_{\mathrm{AXC}} \leq \mathrm{RAT}_{\mathrm{ABC}} \tag{3}$$

As a result of these improvements, MSCT/U is adaptive to delays which may occur during the periodic tours. Contrary to the MSCT method in which tour is static, MSCT/U creates tours adaptive to delays. In the following section, to observe the success of the proposed MSCT/U method, simulation environment and results of experiment tests are provided.

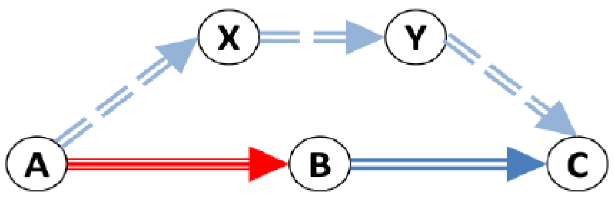

Fig. 1. Improving the created route.

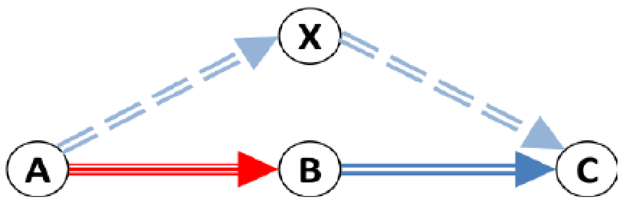

Fig. 2. Shortening the created route.

TABLE I: SIMULATION PARAMETERS

Parameter	Default Value	Range	Note
Topology	PR76		TSP benchmark file
Simulation Time	(Memory Capacity / Sensing Rate) ×20		
Warm-up Period	(Memory Capacity / Sensing Rate) ×3		
Sensing rate	1B/s		
Memory capacity	32KB	4-32	
MS speed	32 Km/h	4-32	
α	2		Tolerance for possible delay
γ	1000	2000	Total delay parameter

III. SIMULATION AND EXPERIMENT RESULTS

WSN is simulated using MASON discrete event multi-agent simulation library [4]. Table I summarizes important simulation parameters and their default values. In simulations, network topology is created according TSPLIB benchmark problem given in [5].

To model the uncertainty in travelling time between SNs we follow the approach proposed by Bertsimas and Sim in [6]. Uncertainty of the traveling times between nodes is bounded to vary within intervals. Specifically, the uncertainty intervals for the travel time t_{ij} between sensor nodes SN_i and SN_j are expressed with minimum travel time ($\mathrm{min}t_{ij}$) and maximum travel time ($\mathrm{max}t_{ij}$). That is,

$$\mathrm{min}t_{ij} \leq t_{ij} \leq \mathrm{max}t_{ij} \tag{4}$$

Minimum travel time is the time to travel between nodes without any delay. On the other hand, maximum travel time is assumed to be double of the minimum travel time. Thus, without any delay traveling time equals to minimum travel time. The maximum delay can cause travelling time to be equal to maximum travel time. Delays are assigned randomly to travelling times between node pairs according to a delay ratio (ϑ) as in Eq. 5. Thus, the maximum amount of delay (δ) is limited by the minimum travel time.

$$\delta_{ij} = \vartheta \times mint_{ij} \quad s.t. 0 \leq \vartheta \leq 1 \tag{5}$$

Thus, travel time is formulated as in Eq. 6.

$$RAT_{ij} = \text{mint}_{ij} + \delta_{ij} \qquad (6)$$

Moreover, to limit the total amount of uncertainty in the traveling times, a total delay parameter (γ) is used. That is,

$$\textstyle\sum_{i,j \in A} \delta_{ij} \leq \gamma \qquad (7)$$

For the given total delay parameter (γ), simulation test scenarios are created as follows. First, Euclidean distances between the given coordinates in the benchmark file are calculated, and the minimum traveling times between SNs are determined according to the given MS speed. Using random ϑ values and given γ value, traveling times between randomly selected nodes are set.

Each experiment of simulation is run 76 times by selecting one of the coordinates in the PR76 benchmark file as the starting node. The observed overflow numbers and amount of collected data are averaged. Thus, as performance metrics, average number of overflows and average amount of data collected are used.

To compare the success of the MSCT/U method, MSCT is implemented. Furthermore, to find the possible best results, we implemented a new method called MSCT/Oracle that knows the exact delays beforehand. Thus, MSCT/Oracle produces best possible solution which can be achieved by MSCT for the given delays on the topology.

Below, the results of experiments are given and discussed.

A. Results for Lower Delay Probability

In these experiments, to observe the effect of a lower delay probability, total delay parameter (γ) value is set 1000. In the first experiments, the results are obtained for different SN memory capacity (see Fig. 3 and Fig. 4). In Fig. 5 and Fig. 6, the results for different MS speed values are given.

As seen in Fig. 3, MSCT/U is very successful at minimizing the overflow incidents. As expected, MSCT/Oracle achieves the minimum overflow numbers for all the different memory capacities as it knows all the delay before they occur. Results of MSCT/U are very close to these optimum values whereas original MSCT is far worse compared to MSCT/U. For the largest memory capacity (32KB), all the methods results with no overflow at all, because MS has abundant time to visit all SNs in the topology. On contrary, when sensors have very small memory capacity (4KB), all proposed methods produce maximum number of overflows due to the fact that MS has very little time to cover a larger number of SNs. For the intermediate memory capacity values, methods can show their success.

Fig. 4 displays the amount of data collected for various memory capacity values. In this figure, it is observed that MSCT/U can generate results as good as the optimum method MSCT/Oracle does. As seen in Table II, MSCT/U can collects more data compared to MSCT due to the fact that MSCT/U prevents sensor memories from overflows successfully. For example, when sensor memory capacities are only 4 KB, MSCT/U collects almost double amount of data compared to MSCT. Moreover, MSCT/U collects about the same amount of data as MSCT/Oracle for all memory levels.

Fig. 5 and Fig. 6 depict the results obtained for various MS speeds. Fundamentally, these results are in parallel with the previous expectations. MSCT/U again produces very close results to the optimum values generated by MSCT/Oracle for various MS speeds. For instance, as seen in Fig. 6, when MSCT/U is applied, collected data for various MS speed values are, on the average, 10% less than the optimum values, but compared to the original MSCT, up to 80% more.

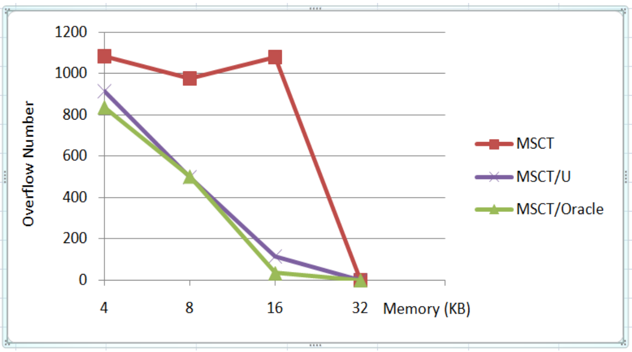

Fig. 3. Overflow numbers for different memory capacities (γ=1000).

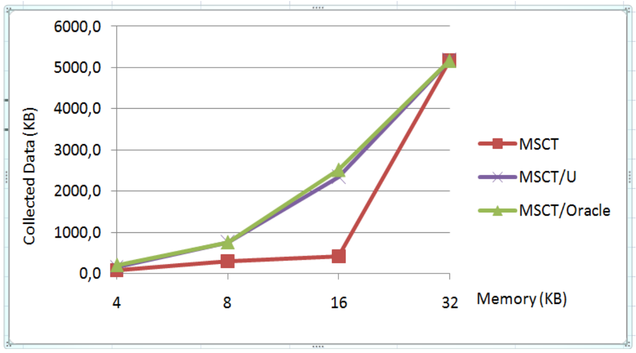

Fig. 4. Amount of collected data for different memory capacities (γ=1000).

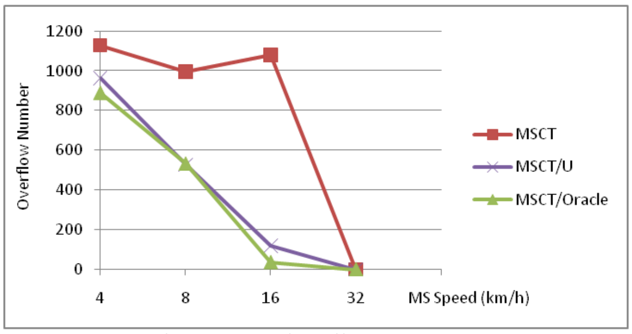

Fig. 5. Overflow numbers for different MS speeds (γ=1000).

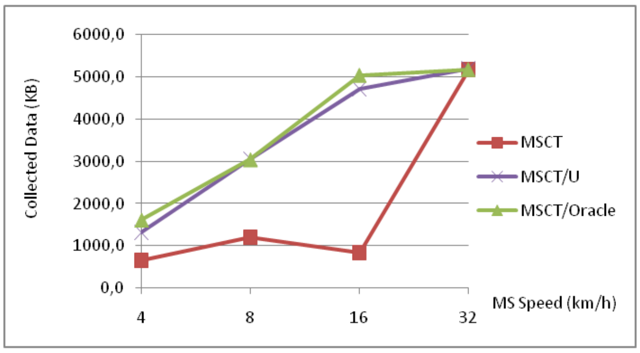

Fig. 6. Amount of collected data for different MS speeds (γ=1000).

The above observations show that MSCT/U is robust for changes in important WSN parameters such as sensor memory capacity and MS speed. In the following tests, the effect of increased uncertainty and delay on the success of the proposed method is investigated.

TABLE II: AMOUNT OF COLLECTED DATA FOR DIFFERENT MEMORY CAPACITIES (Γ=1000)

Memory (KB)	MSCT	MSCT/U	MSCT/Oracle
4	82.3	161.0	202.1
8	299.5	763.8	766.0
16	418.7	2351.7	2515.8
32	5166.1	5180.9	5183.2

B. Results for Higher Delay Probability

In these set of experiments, total delay parameter (γ) value is increased to 2000 which means higher amount of delays and uncertainty exist in the topology. The results of these experiments are given in Fig. 7 and Fig. 8 for different MS speeds, and in Fig. 9 and Fig. 10 for different sensor memory capacities. In these results, MSCT/U generates successful schedules to deal with higher uncertainty. As MSCT/Oracle knows all the delays, it generates the optimum solutions. Compared to MSCT/Oracle, MSCT/U discovers delays and adapts the tour considerably well. Results show that MSCT/U can handle increased uncertainty so well that it produces results very similar to the optimum values. Thus, as seen in all figures, MSCT/U improves MSCT as expected. Without the improvements, MSCT performs far worse than the optimum solution.

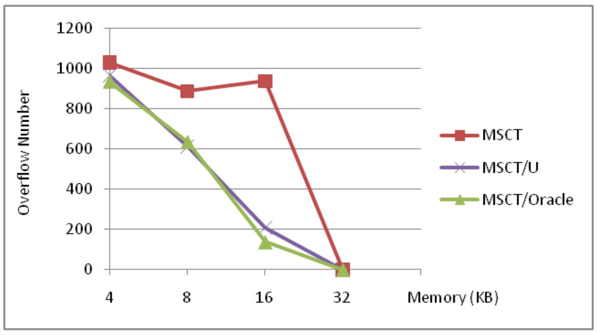

Fig. 7. Overflow numbers for different memory capacities (γ=2000).

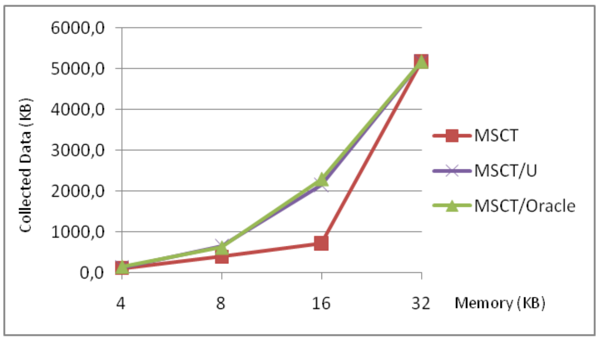

Fig. 8. Amount of collected data for different memory capacities (γ=2000).

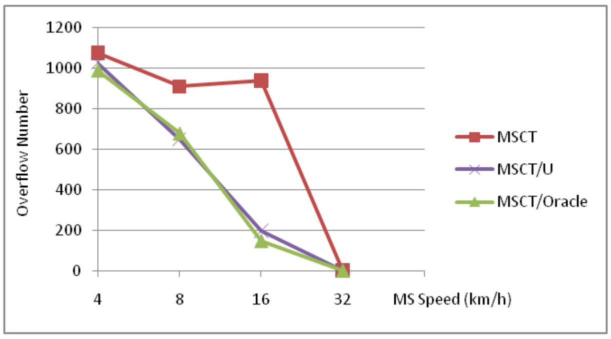

Fig. 9. Overflow numbers for different MS speeds (γ=2000).

Moreover, when all results are compared, it is observed that change in total delay parameter (γ) value, that is uncertainty, does not deteriorate the performance of MSCT/U. These results show the robustness of the proposed improvements under different levels of uncertainty.

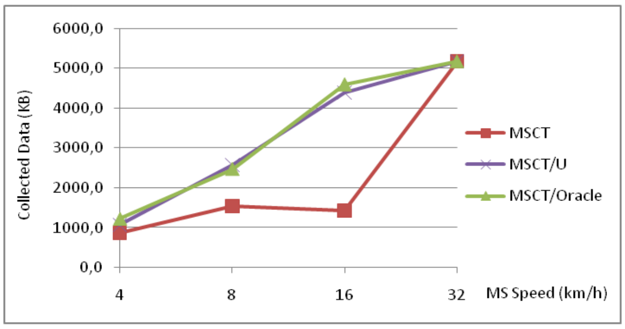

Fig. 10. Amount of collected data for different MS speeds (γ=2000).

IV. CONCLUSION

Uncertainty is inevitable in real life applications. In WSN, data gathering by mobile sink faces uncertainty in travelling times between sensor nodes. To deal with uncertainty, MSCT/U is developed by improving the MSCT method. Furthermore, to calculate the optimal solution for the given uncertainty, MSCT/Oracle is implemented. The results show that MSCT/U generates schedules as good as the optimum method does. Furthermore, MSCT/U displays its robustness when tested against different MS speed, sensor memory, and uncertainty values.

As a future work, we plan to deal with uncertainty in the collected data together with the travelling time. In addition, we would like to create a method to schedule more than one mobile sink.

REFERENCES

[1] A. Somasundara, A. Ramamoorthy, and B. Srivastava, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," in *Proc. International Real-Time Systems Symposium*, 2004.

[2] A. Somasundara, A. Ramamoorthy, and B. Srivastava, "Mobile element scheduling with dynamic deadlines," *IEEE Transactions on Mobile Computing*, vol. 6, no. 4, pp. 395-410, 2007.

[3] M. Karakaya, "MSCT: An efficient data collection heuristic for wireless sensor networks with limited sensor memory capacity," 2015.

[4] S. Luke, C. Revilla, L. Panait, and K. Sullivan, "Mason: A new multi-agent simulation toolkit," in *Proc. SwarmFest Workshop*, 2004.

[5] TSPLIB web site. [Online]. Available: http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/

[6] D. Bertsimas and M. Sim, "Robust discrete optimization and network flows," *Mathematical Programming*, vol. 98, no. 1-3, pp. 49-71, 2003.

Murat Karakaya received the B.S.E.E. degree in 1991 from the Turkish Military Academy (KHO), Ankara, Turkey, and the M.S. and Ph.D. degrees in computer engineering from Bilkent University, Ankara, Turkey in 2000 and 2008, respectively. From 1992 to 2000, he worked as an engineer at different units in the Turkish Land Forces (KKK), Ankara, Turkey. From 2000 to 2005, he worked as an instructor and software engineer at the Turkish Military Academy (KHO), Ankara, Turkey. Then, during 2005 to 2008, he worked as an IT project manager in the North Atlantic Treaty Organization (NATO) Brussels, Belgium. From 2008 to 2012, he worked as an instructor and software engineer at the Turkish Military School of Electronics, Communications and Information Systems (MEBS) and Turkish Military Academy (KHO), Ankara, Turkey. He joined the Faculty of Atilim University in 2012 and is currently an Asst. Prof. in the Department of Computer Engineering, Ankara, Turkey. His research interests are natural computing, sensor networks, peer-to-peer networks, natural computing, optimization, and communications protocol design.