uScan: A Lightweight Two-Tier Global Sensing Coverage Design

Yu Gu and Tian He
Department of Computer Science and Engineering, University of Minnesota
{yugu,tianhe}@cs.umn.edu

1 Introduction

Wireless Sensor Networks (WSNs), consisting of thousands of low-cost sensor nodes, have been used in many application domains such as military surveillance [1], habitat monitoring [4] and scientific exploration [6]. Limited power supplies and difficulties in harvesting ambient energy make energy conservation a critical issue to address. As one of solutions, energy-efficient sensing coverage extends system lifetime by leveraging on the redundant deployment of sensor nodes. Existing algorithms [5, 7, 8, 9] are designed to be well distributed and localized with solid performance gains. While the state-of-the-art is encouraging, we believe there are some aspects that need further investigation. In most algorithms, extending system lifetime is achieved essentially through coordination among neighboring nodes. The local node density, therefore, imposes a theoretical upper bound on the system lifetime, if a continuous sensing coverage or a partial coverage is required. Such a bound can be surpassed through global scheduling. However, the overhead of global scheduling would increase significantly if the coordination among the nodes goes beyond the neighborhood.

To address these issues, we introduce a two-tier global scheduling method, called uScan. At the first level, coverage is scheduled to activate different portions of an area. We propose an optimal scheduling algorithm to minimize area breach. At the second level, sets of nodes are selected to cover active portions. Interestingly, we show that it is possible to obtain optimal set-cover results in linear time if the layout of areas satisfies certain conditions. We have implemented and evaluated our design on the Berkeley TinyOS/Mote platform [2], using 30 MicaZ motes. The results indicate that uScan is a lightweight solution with significant energy savings, compared with localized solutions.

2 uScan Design

uScan is a two-level schedule algorithm, which works as follows: Suppose we provide sensing coverage to a given area using uScan. First, uScan divides the area into small regions, and decides the working schedules for these regions. This level of scheduling is conceptually independent of the deployment of the nodes. At the second-level, we assign nodes to cover the active regions at different time intervals, using a set-cover technique. By combining the first-level schedule and the set-cover assignment, we can decide the working schedule of individual nodes.

2.1 Assumption

We assume that nodes are time-synchronized and their locations are known. These are common assumptions for many sensor network applications [1, 4, 6]. Accuracy of time synchronization and localization do not need to be precise, because clock drift can be resolved by slightly extending the active duration and localization error can addressed using the method proposed in [8]. For the clarity of the protocol description in the rest of the paper, we refer the sensing area of a node as a circle with a nominal radius r centered at the location of the node. However, our design works under irregular sensing areas as long as nodes are aware of their sensing areas.

2.2 Definition of the Node Schedule

In essence, a sensing coverage algorithm decides the working schedule of individual sensor nodes. Specifically, uScan describes the behavior of a node using two parameters, namely the schedule bits S and switching rate R.

- **Schedule bits S:** It is an infinite binary string in which 1 denotes the active state and 0 denotes the inactive state. Since the sensing coverage schedule is usually periodic, follows a certain pattern. Therefore, we can express S with a regular expression. For example, $(0010)^*$ can be used to denote a repeated off-off-active-off schedule.
- **Switching rate R:** It defines the rate of toggling between states. For example, a switching rate of 0.5HZ requires a node to read one bit from the schedule S every 2 seconds (when consecutive bits have the same value, there is no need to change power state of the node physically).

2.3 Level I: Tile Scheduling

In uScan, we partition an area under surveillance into some small regions of the same shape, a process called tessellation. These small regions are called *tiles*, which can be regular triangles, rectangles or regular hexagons in a 2-D space. The size of tiles is set to be smaller than the minimum target size, so that a target is detected as long as a portion of a tile is covered. In this section, we discuss two methods for the tile-level scheduling. They differ in the energy consumption rate and the detection delay.

- **Line Scan:** Instead of trying to cover all tiles, we only cover a column/row of tiles in a certain interval of time during one round of scan. The covered columns/rows are increasing or decreasing consecutively.
- **Systolic Scan:** Systolic Scan emulates the cardiac cycles of a beating heart. Over the area under surveillance, we sense the tiles from the inner layer to the outer layer continuously.

Both line scan and systolic scan specify only the set of tiles need to be activated (covered) at a given point of time. The task of covering each tile set is accomplished by the second-level node scheduling, which is described in the next section.

2.4 Level II: Node Scheduling

Tile-level scheduling determines the set of active tiles TS_i at the time interval i. In this section, we describe how we can

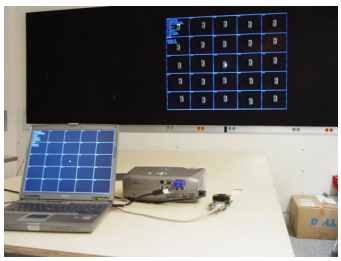

Fig. 1. Test-bed Setup

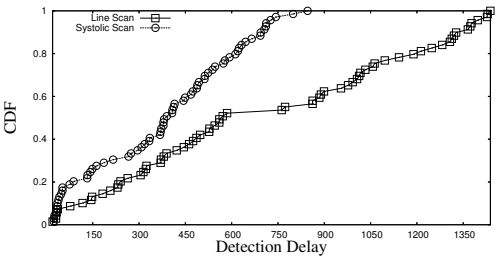

Fig. 2. Detection Delay

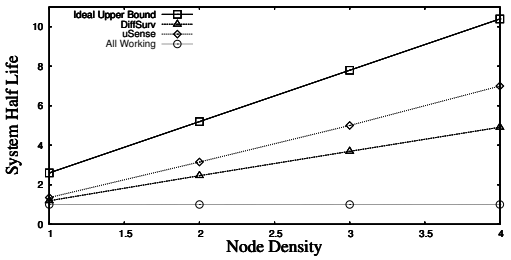

Fig. 3. System Half Life vs. Densities

translate a known tile schedule into a corresponding node schedule bits S, which can be interpreted directly by a generic switching algorithm.

The main idea of our approach is to find the optimal set of nodes which could cover all the tiles that need to be active at time interval i. Before node scheduling, we first map physical node coverage into the coverage bipartite graph according to the coverage relationship. Then we divide node scheduling into two steps. First, for a tile set TS_i, we keep identifying 1-cover set with minimal number of nodes, until the size of 1-cover set is above a certain threshold. Secondly, we create schedules for nodes such that each identified 1-cover set provides coverage to TS_i in a round-robin fashion.

The generic Minimum Set Cover problem has been proven NP-Hard [3]. Fortunately, we find line scan coverage is a special case of the generic set cover problem, because a node only needs to cover a continuous segment of tiles. By mapping the coverage bipartite graph into a directed acyclic graph with following rules:

1. Map N tiles in TS_i into N vertex $V = \{v_1, ..., v_N\}$ and add one extra vertex v_{N+1}.
2. If a node covers a set of tiles $\{T_i, ..., T_{i+n}\}$, we create n directional edges (v_i, v_j) where $v_j = v_{i+1}, ..., v_{i+n+1}$. Each edge has a unit cost.

We reduce the tile set cover problem to the problem of finding out the shortest paths from v_1 to v_{N+1}, with the overall runtime of $O(|V|) + O(|E|)$.

We note that the proposed polynomial algorithm does not apply to generic tile scheduling, where a tile set does not form a continuous curve or where a node can cover multiple segments of a tile set simultaneously. In these cases, we adopt a greedy set-cover method by choosing the node that covers the most number of tiles first.

In order to support line scan or systolic scan in a 2-D space, we need to identify cover sets for the whole area (not just for a single tile set). Thus a node may need to cover multiple tile sets TS_i. To effectively handle the cases where we have to select cover sets for multiple TS, we designed an algorithm that each node maintains a counter SC which records how many times it has been selected into a unique cover sets. While selecting cover sets for each tile set TS, instead of solely consider the number of nodes in the cover sets, the algorithm calculates the minimum cover set among the nodes whose SC counter values are as small as possible.

After obtaining cover sets for every tile set TS_i, we build the final schedule of node according to all the cover sets it's belonged to, which can be executed directly by our generic switching algorithm.

3 Implementation and Evaluation

We have implemented a complete version of uScan on Berkeley TinyOS/Mote platform, using 30 MicaZ motes as shown in Figure 1. The compiled image of a full implementation occupies 21,040 bytes of code memory and 907 bytes of data memory. The results showed that uScan is a lightweight and efficient coverage design. To reveal the system performance at scale, we have conducted some initial large scale simulations with 10,000-node. Under full coverage mode as shown in Figure 3, we demonstrated that our global scheduling algorithms provide significant energy savings over previous protocols such as DiffSurv [8] under metrics such as Half-life, Coverage Overtime and Node Energy Consumption. In the future, we plan to investigate the performance under partial coverage mode at scale as well, with additional metrics such as Detection Delay for Static Targets and Worst-Case Breach (WCB) for Mobile Targets.

4 Conclusion

The major contribution of this work is a two-level global scheduling algorithm called uScan. In the first level, we propose two tile-level scheduling algorithm. In the second level, we propose a linear algorithm to address the set-cover problem when the layout of tiles satisfies certain conditions. We evaluate our architecture with a network of 30 MicaZ motes, an extensive simulation with 10,000 nodes, as well as theoretical analysis. We believe our work has successfully provided flexibility and efficiency for the sensor network coverage problem.

5 References

[1] A. Arora and et al. A Wireless Sensor Network for Target Detection, Classification, and Tracking. *Computer Networks (Elsevier)*, 2004.

[2] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System Architecture Directions for Networked Sensors. In *ASPLOS'00*, 2000.

[3] V. T. Paschos. A Survey of Approximately Optimal Solutions to Some Covering and Packing Problems. In *ACM Computing Surveys*, June 1997.

[4] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler. An Analysis of a Large Scale Habit Monitoring Application. In *SenSys'04*, 2004.

[5] D. Tian and N. Georganas. A Node Scheduling Scheme for Energy Conservation in Large Wireless Sensor Networks. *Wireless Communications and Mobile Computing Journal*, May 2003.

[6] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, S. Burgess, D. Gay, P. Buonadonna, W. Hong, T. Dawson, and D. Culler. A Macroscope in the Redwoods. In *Sensys'05*, November 2005.

[7] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. In *Sensys'03*, November 2003.

[8] T. Yan, T. He, and J. A. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *SenSys'03*, November 2003.

[9] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks. In *Proc. of International Conference on Distributed Computing Systems (ICDCS)*, May 2003.

uScan: A Lightweight Two-Tier Global Sensing Coverage Design

Yu Gu and Tian He

Department of Computer Science and Engineering, University of Minnesota

{Yugu, tianhe}@cs.umn.edu

Research Issue:

Wireless Sensor Networks (WSNs), consisting of thousands of low-cost sensor nodes, have been used in many application domains such as military surveillance, habitat monitoring and scientific exploration. Limited power supplies and difficulties in harvesting ambient energy make energy conservation a critical issue to address. As one of solutions, energy-efficient sensing coverage extends system lifetime by leveraging on the redundant deployment of sensor nodes. Existing algorithms are designed to be well distributed and localized with solid performance gains. While the state-of-the-art is encouraging, we believe there are some aspects that need further investigation. In most algorithms, extending system lifetime is achieved essentially through coordination among neighboring nodes. The local node density, therefore, imposes a theoretical upper bound on the system lifetime, if a continuous sensing coverage or a partial coverage is required.

Project Overview

To address these issues, we introduce a two-tier global scheduling method, called uScan. At the first level, coverage is scheduled to activate different portions of an area. We propose an optimal scheduling algorithm to minimize area breach. At the second level, sets of nodes are selected to cover active portions. Interestingly, we show that it is possible to obtain optimal set-cover results in linear time if the layout of areas satisfies certain conditions. We have implemented and evaluated our design on the Berkeley TinyOS/Mote platform [2], using 30 MicaZ motes. The results indicate that uScan is a lightweight solution with significant energy savings, compared with localized solutions.

Assumptions:

We assume that nodes are time-synchronized and their locations are known. These are common assumptions for many sensor network applications. Accuracy of time synchronization and localization do not need to be precise, because clock drift can be resolved by slightly extending the active duration and localization error can addressed using the method proposed by DiffSense. For the clarity of the protocol description, we refer the sensing area of anode as a circle with a nominal radius centered at the location of the node. However, our design works under irregular sensing areas as long as nodes are aware of their sensing areas.

Asymmetric Architecture:

We employ an asymmetric architecture to improve the flexibility and extensibility of the design.

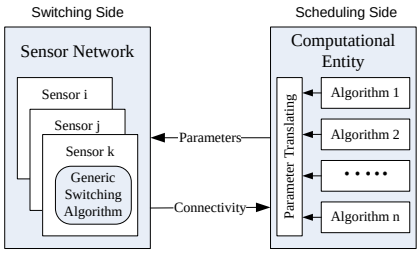

Fig 1. Asymmetric System Architecture

Specification of Node Schedule:

In essence, a sensing coverage algorithm decides the working schedule of individual sensor nodes. Specifically, uScan describes the behavior of a node using two parameters, namely the schedule bits S and switching rate R.

Schedule bits S: It is an infinite binary string in which 1 denotes the active state and 0 denotes the inactive state.

Switching rate R: It defines the rate of toggling between states.

Tier I: Tile Scheduling:

Tile-level scheduling determines the set of active tiles at a certain time interval. Nodes within a sensor network only support a generic switching algorithm, which has neither the concept of tiles nor the partition information of the tiles.

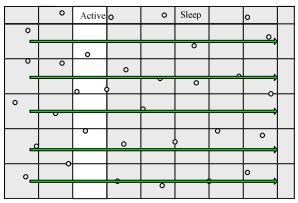

Fig 2. Line Scan

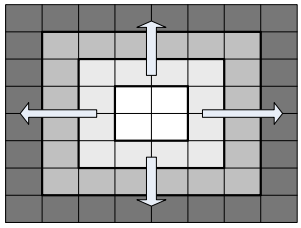

Fig 3. Systolic Scan

Tier II: Node Scheduling

Node Scheduling translates a known tile schedule into a corresponding node schedule bits S. The main idea of our approach is to find the optimal set of nodes which could cover all the tiles that need to be active at time interval i. Before node scheduling, we first map physical node coverage into the coverage bipartite graph according to the coverage relationship. Then we divide node scheduling into two steps. First, for a tile set TS_i, we keep identifying 1-cover set with minimal number of nodes, until the size of 1-cover set is above a certain threshold. Secondly, we create schedules for nodes such that each identified 1-cover set provides coverage to TS_i in a round-robin fashion.

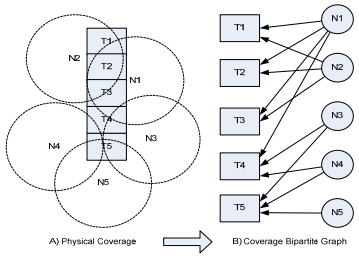

Fig 4. Set-Cover Based Scheduling

Implementation:

We have implemented a complete version of uScan on Berkeley TinyOS/Mote platform, using 30 MicaZ motes as shown below. The compiled image of a full implementation occupies 21,040 bytes of code memory and 907 bytes of data memory.

Fig 5. System Implementation

Evaluation:

We evaluate our architecture with physical system as well as an extensive simulation with 10,000 nodes.

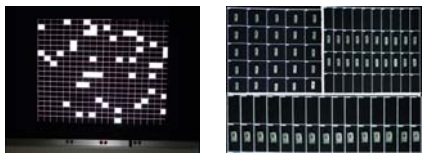

Fig 6. Evaluation Set-up

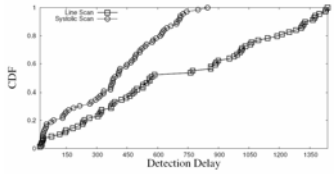

Fig 7. Detection Delay

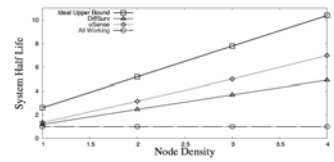

Fig 8. System Life-Time over Different Node Densities

Conclusion:

The major contribution of this work is a two-level global scheduling algorithm called uScan. In the first level, we propose two tile-level scheduling algorithm. In the second level, we propose a linear algorithm to address the set-cover problem when the layout of tiles satisfies certain conditions. We evaluate our architecture with a network of 30 MicaZ motes, an extensive simulation with 10,000 nodes, as well as theoretical analysis. We believe our work has successfully provided flexibility and efficiency for the sensor network coverage problem.

Acknowledgements:

This work is supported by NSF Nets NOSS Program

More information can be found at
Minnesota Embedded Sensor System Group
Http://mess.cs.umn.edu