# A Gossip-enabled Spatial Index for Moving Object Management over Wireless Sensor Networks*

Chao-Chun Chen
Dept. of Comp. Sci. & Info. Engr.
Southern Taiwan University
chencc@mail.stut.edu.tw

Chiang Lee
Dept. of Comp. Sci. & Info. Engr.
National Cheng-Kung University
leec@mail.ncku.edu.tw

Yen-Liang Liu
Dept. of Comp. Sci. & Info. Engr.
National Cheng-Kung University
liuyl@ads.csie.ncku.edu.tw

## Abstract

*More and more users or applications showed high interest in obtaining moving object locations from a wireless sensor network. However, a moving object usually updates its location very frequently, and thus, a great number of messages for updating the locations in the wireless sensor networks are incurred. In this paper, we propose an index structure, called the* Gossip-enabled Spatial Index Tree (GSI-tree)*, to support efficient communication for location updates. Our idea is to utilize the gossip property of the sensor communication to let the neighboring paths of the GSI-tree share the transmitted message. In this way, the nodes managing a past location can deregister the object location independently by hearing the registration messages from the neighboring nodes.*

## 1 Introduction

Recent advances in the wireless sensor network have led to a vast and expeditious development for location-based applications [9, 3]. For example, when a disaster happens, a fixed communication system (e.g., GSM or WIFI) may be out of work due to the damage of the infrastructure. In this situation, employing wireless sensor nodes is a practical solution to rapidly build ad-hoc communication networks, and the emergency system can then arrange human resource or activate other related functions according to the conditions observed from the wireless sensor network. Through reporting current locations of all users in the sensing field, the emergency system can guide victims to safety places and aid workers to those places that need most help. Moreover, the emergency system can also help the aid workers to search for the wounded in their neighborhood by injecting a range query or a nearest neighbor query to the wireless sensor network.

In the above scenarios, the critical technique for supporting location-based applications is the location management of mobile users in the wireless sensor network. In this paper, we propose a communication-efficient index structure, called *Gossip-enabled Spatial Index Tree (GSI-Tree)*, to manage the locations of moving objects in a wireless sensor network. GSI-tree is able to reduce the communication cost for updating moving object locations. Its idea is to utilize the *gossip property* (to be detailed later) of sensor communication in the construction of GSI-tree. The GSI-tree is best for a sensor network with the deployment of at least one-coverage, that is, every spot within the sensor network is monitored by at least one sensor [6]. The GSI-tree is built in such a way that each node and its left (right) sibling should be within a one-hop communication range as long as possible.

Figure 1 is an example to illustrate the benefit of GSI-tree. Most location management schemes has two operations for managing object locations[7, 5]. One is the location registration operation in the new node, and the other is the location deregistration operation in the old node. Figure 1(a) shows the movement of the object from node A to node B. Six communication messages are needed in this case to update the object location in the tree. Three messages are needed for location registration (bold arrows in the figure) and three messages are needed for location deregistration (thin arrows in the figure). On the other hand, Figure 1(b) indicates the movement of the object in the GSI-tree. In this case, only three messages are needed for location registration (bold arrows in the figure). This is because the left (right) sibling of a node is within a one-hop range in the GSI-tree and the communication of in the sensor network is a broadcast to all neighbors [1]. Hence, the nodes that involved in a location deregistration operation can be informed by listening to the broadcast of neighbors that performing location registration operation (dashed arrows in the figure). Hence, only three messages are needed for loca-

*This work was supported by National Science Council of Taiwan (R.O.C) under Grants NSC96-2221-E-006-261-MY2, NSC96-2221-E-218-050-MY2, and NSC96-2515-S-218-002.

978-0-7695-3457-2/08 $25.00 © 2008 IEEE
DOI 10.1109/MSN.2008.19

tion registration and no additional messages are needed for location deregistration. In a high-density sensor network, a GSI-tree-based moving objects management policy needs only about one half of the communication cost that is required in other methods.

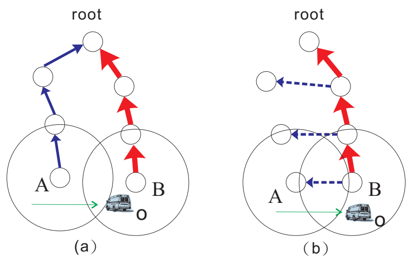



**Figure 1. (a)general tree (b)GSI-tree**

The remainder of this paper proceeds as follows. We describe the system architecture in Section 2. We then presents the construction method of a GSI-tree in Section 3. Section 4 presents the update process and search process in a GSI-tree. Finally, we conclude the paper in Section 5.

## 2 System Architecture

A wireless sensor network consists of hundreds or thousands of sensor nodes spreading in a sensing field. Each sensor node has operating components: sensing, storage, computation, and wireless communication [1]. Through these components, distributed sensors are able to cooperatively track a moving object, and store its location distributively in the sensors [8]. If the distance between two sensor nodes is longer than the antenna's communication range, data delivery between the sensors will have to be done by using a multi-hop routing technique, e.g., GPSR. The sensor application communicates with clients from internet through a gateway, such as receiving a query and returning the requested data. In a wireless sensor network, the communication protocol of a sensor is basically broadcast-based [1]. Hence, all sensors within the communication range can hear the broadcast message. This characteristic is referred to as the *gossip property* in sensor networks [2, 4].

## 3 Gossip-enabled Spatial Index Tree

*Gossip-enabled Spatial Index Tree (GSI-Tree)* is a distributed tree structure with the gossip property of wireless sensor networks, and is designed to reduce update cost for the movements of moving objects. In traditional methods for managing object movement, two fundamental operations, deregistration of previous location and registration of the new location, are performed [5]. Thus, two types of communication cost are counted for object movement. The key idea of our method is to let the sibling nodes of the GSI-tree as close to each other as possible so that the deregistration operations for the management of a moving object become unnecessary. In this way, the communication cost may be reduced to one half of the cost in traditional methods. The construction of this GSI-tree is presented in the following.

The sensor network is initialized as follows. It is divided into different levels by concentric squares, as shown in Figure 2. Assume the root is in the center of the network, and the shortest distance between the root and each edge of a square is $c \times d$, where $d$ is the distance between two levels and $c$ is the level number. In this sensor network, the inner most region is called the level-1 region, and sensors in the level-1 region are labeled as level-1 sensors. Since the distance between the center of the sensor network and the edge of the level-1 region is $d$, the edge length of the level-1 region is equal to $2 \times d$. Following this manner, the level of other regions can be determined according to their distances to the center. For example, the level-$k$ region is the region between the square with edge length $k \times 2d$ and the square with edge length $(k-1) \times 2d$.

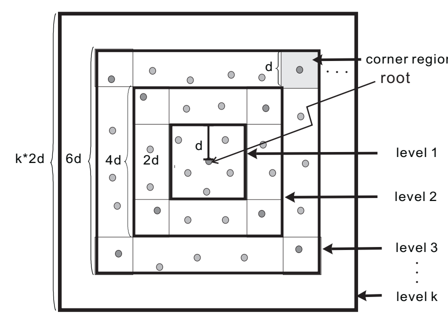



**Figure 2. Illustration of level-$k$ region.**

In general, a tree structure consists of nodes and edges. For a GSI-tree, the tree nodes are the subset of the sensor nodes and the tree edges are the communication path of tree nodes. The construction of the GSI-tree is to first find tree nodes, and then to create edges for the tree nodes. The construction includes four steps, and is described as follows.

**Step 1: Initialization**

The first step is initialization, that is, setting a level number for each node according to the level they reside. Thus, a sensor node in the level-$k$ region sets its level number to $k$ (that is, level $k$).

**Step 2: Determine corner sensors**

The second step is to determine the four corner sensors of each level. The corner sensors are the starters that are used in Step 4 to find the GSI-tree nodes of each edge. The corner sensors for a level can be determined by the following two substeps. The first substep is to choose the candidates of the corner sensors. In order to find the candidates, a rectangle, called the *corner region*, needs to be determined for each of the four corners. The corner region is the square area that is at one of the four corners of each level, and the edge length of a corner region is equal to the distance between two consecutive levels, i.e., $d$. An example of a corner region is given in Figure 2 as a shadow square. Only a sensor node whose detecting area covers the corner region can be a candidate of the corner sensors. The second substep is to determine the corner sensor for each corner. The corner sensor is the one that has most number of neighbors among candidates in the same corner. Selecting the node with the most number of neighbors as a corner sensor is because more relationships of a node increase the success probability of finding a tree node in the next step.

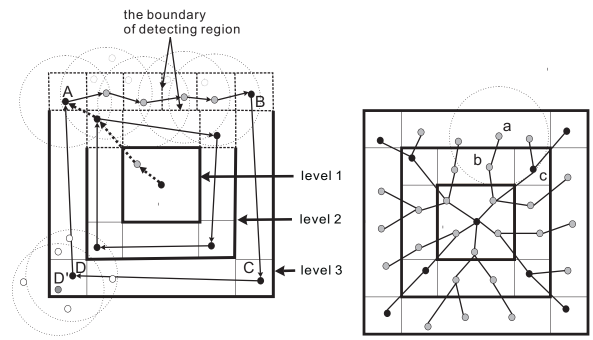
Figure 3(a) shows an example of this step. Both sensor $D$ and sensor $D'$ are in the bottom-left corner region of level 3, and have six and four neighbors, respectively. As sensor $D$ has more neighbors than sensor $D'$ has, sensor $D$ is a corner sensor of level 3.

**Step 3: Determine tree nodes**

After finding the corner sensors of each level in the second step, we move on to select the tree nodes. Tree nodes are those nodes that will be used to build a tree to the root. The root starts this step by sending a "tree-node finding" message to the upper-left corner sensors at each level. These corner sensors are the "seed" tree nodes, meaning that they themselves are tree nodes and from them we search for other tree nodes. It proceeds in this way. Each corner sensor finds from its clockwise neighbors at the same level the one that is closest to it, and set this node as a tree node. Then, this tree node subsequently finds the next tree node in the same manner. The process continues until all tree nodes at the same level are all determined. An example of finding the tree nodes is given in Figure 3(a), in which the tree nodes from corner sensor A to corner sensor B are shown and each being pointed by a counterclockwise neighbor. Note that these arrowed edges between the tree nodes in Figure 3(a) will not be an edge of the future GSI-tree. The edge of the GSI-tree will be an edge connecting two tree nodes at different levels. The reason that we select tree nodes in the above manner (choosing the one closest to another tree node) is to ensure that the aforementioned gossip property holds in the GSI-tree. At the end of this step, all tree nodes of the GIS-tree are determined (but the edges are not yet).

**Step 4: Determine tree edges**

The fourth step is to determine tree edges, that is, finding the parent for all tree nodes. In this step, a tree node at level $i$ would choose as parent the node that is a directly communicable with the node at the level $(i-1)$. If there exists multiple directly communicable GSI-tree nodes, the parent would be the node whose number of children is minimal. The purpose is to balance the energy consumption of the sensor network and prolong the network life time. Figure 3(b) shows an example of a finished GSI-tree. In the figure, sensor $a$ at level 3 can directly communicate with sensors $b$ and $c$ at level 2. Since sensor $c$ has two children and sensor $b$ has no child (other than sensor $a$), sensor $a$ thus selects sensor $b$ as its parent. After the above four steps, the GSI-tree is constructed.




(a) Step 2 - Step 3. (b) Step 4.

**Figure 3. Illustration of Step 2 - Step 4.**

The GSI-tree bears two properties. One is that the sibling nodes can share the messages, which is the gossip property. This property is built in Step 3. The other is that the communication of nodes between two consecutive levels can be one-hop due to the setting of the distance between two levels. This property is built in Step 4. By having the above two properties, the GSI-tree is able to manage object locations with minimal cost in communication.

## 4 Update and Search in GSI-tree

### 4.1 Update Operation

An update operation occurs when an object moves across the detecting boundary of two neighboring sensors. When a sensor detects the arrival of an object, the sensor registers the object location and sends an update message to the root along the GSI-tree path. The sensor which detects the leave of an object would deregister the object location, that is, delete the obsolete object location or links. The algorithm shows the deregistration is accomplished by taking advantage of the gossip property in GSI-tree.

Assume the level that an object previously stayed is $k$. A movement can cause three possible cases in a GSI-tree: (1) still at level $k$ but arrives a region monitored by another sensor, (2) from level $k$ to level $(k+1)$, (3) from level $k$ to level $(k-1)$. In the first case, the object moves from the previous node to its sibling node in the GSI-tree. Thus, the new node would send the information of the latest object location to the least common ancestor (LCA) of the new node and the previous node. Since the new node and the previous node are 1-hop neighbors, location update in the previous node and its ancestors can be done by simply broadcasting update message once and then erase the obsolete records.

In the second case, that is, “moving from level $k$ to level $(k+1)$”, location update is processed according to the condition whether the new node is the child of the previous node. In case the new node is a child of the previous node, the new node only updates the latest location to its parent, that is, the previous node. In a GSI-tree, this needs only one communication. In case the new node is not a child of the previous node, the steps are the same to those in the first case. The location update message is transmitted from the new node to the LCA of the new node and the previous node, and the location deregistration is performed in the nodes that maintain the previous location by hearing the transmitted location update message.

In the third case, that is, “moving from level $k$ to level $k-1$”, we need to consider whether the new node is the parent of the previous node. In case the new node is the parent of the previous node, the new node only broadcast a message to its child (i.e., the previous node) to erase the obsolete record. In case the new node is not the parent of the previous node, the new node would send the latest object location to the LCA of the new node and the previous node. Location update in this case is the same as that in the first case. Notice that there is a slight chance that the previous node at level $k$ might not hear the registration message from the new node at level $k-1$, because a GSI-tree does not promise the gossip property over nodes of different levels. Thus, after the parent of the previous node hears the location update message from the new node, it also needs to broadcast an additional message to erase the obsolete record in the previous node.

### 4.2 Search Operation

A search operation is used to locate a requested moving object in the sensor network. The search operation based on a GSI-tree is much simpler than the update operation. While the location of an object is requested, the location query is transmitted to the root of GSI-tree. Then, the root forwards the query to the child that contains the current location of the requested object. Next, the location query is delivered to the node that currently has the object in the same manner by following the edges in GSI-tree. After this node obtains the latest location of the requested object, the latest location is then directly sent to the query user through the underlying routing protocol.

## 5 Conclusions

Tracking moving object is one of important applications in wireless sensor networks. However, the locations of moving objects usually change very frequently, and a great number of messages for updating the locations in the wireless sensor networks are incurred. In this paper, we propose an in-network index structure, Gossip-enable Spatial Index (GSI-tree), to manage moving object locations in wireless sensor networks. Our idea is to utilize the gossip property of the sensor communication in the GSI-tree so that the neighboring nodes of GSI-tree can share the transmitted message as possible. In this way, the nodes managing the previous location can run the location deregistration operation alone by hearing the registration messages from the neighboring nodes. Hence, the cost for the location deregistration can be greatly reduced.

## References


[1] L. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, Aug 2002.

[2] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright. Geographic gossip: efficient aggregation for sensor networks. In *the Fifth International Conference on Information Processing in Sensor Networks (IPSN)*, pages 69–76, Nashville, Tennessee, USA, April 2006.

[3] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.

[4] A. V. Kini, V. Veeraraghavan, N. Singhal, and S. Weber. SmartGossip: an improved randomized broadcast protocol for sensor networks. In *the Fifth International Conference on Information Processing in Sensor Networks (IPSN)*, pages 210–217, Nashville, Tennessee, USA, April 2006.

[5] H. T. Kung and D. Vlah. Efficient location tracking using sensor networks. In *2003 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1954– 1961, New Orleans, Louisiana, USA, March 2003.

[6] X.-Y. Li, P.-J. Wan, and O. Frieder. Coverage in wireless ad hoc sensor networks. *IEEE Transactions on Computers*, 52(6):753–763, June 2003.

[7] C.-Y. Lin and Y.-C. Tseng. Structures for in-network moving object tracking in wireless sensor networks. In *First International Conference on Broadband Networks (BROADNETS)*, pages 718–727, San Jose, California, USA, October 2004.

[8] J. Xu, X. Tang, and W.-C. Lee. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Transactions on Parallel and Distributed Systems*, 19:262–275, 2008.

[9] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, 19(2):61–72, March 2002.