# Location Tracking in a Wireless Sensor Network by Mobile Agents and Its Data Fusion Strategies [*]

Yu-Chee Tseng, Sheng-Po Kuo, Hung-Wei Lee, and Chi-Fu Huang

Department of Computer Science and Information Engineering
National Chiao-Tung University
Hsin-Chu, 30050, Taiwan
E-mail:{yctseng, spkuo, leehw, and, cfhuang}@csie.nctu.edu.tw
(corresponding author: Prof. Yu-Chee Tseng)

**Abstract**

The wireless sensor network is an emerging technology that may greatly facilitate human life by providing ubiquitous sensing, computing, and communication capability, through which people can more closely interact with the environment wherever he/she goes. To be context-aware, one of the central issues in sensor networks is *location tracking*, whose goal is to monitor the roaming path of a moving object. While similar to the location-update problem in PCS networks, this problem is more challenging in two senses: (1) there are no central control mechanism and backbone network in such environment, and (2) the wireless communication bandwidth is very limited. In this paper, we propose a novel protocol based on the *mobile agent* paradigm. Once a new object is detected, a mobile agent will be initiated to track the roaming path of the object. The agent is mobile since it will choose the sensor closest to the object to stay. The agent may invite some nearby slave sensors to cooperatively position the object and inhibit other irrelevant (i.e., farther) sensors from tracking the object. As a result, the communication and sensing overheads are greatly reduced. Our prototyping of the location-tracking mobile agent based on IEEE 802.11b NICs and our experimental experiences are also reported.

**Keywords:** ad hoc network, context-aware computing, location tracking, mobile computing, sensor network, wireless communication.

## 1 Introduction

The rapid progress of wireless communication and embedded micro-sensing MEMS technologies have made *wireless sensor networks* possible. Such environments may have many inexpensive wireless nodes, each capable of collecting, processing, and storing environmental information, and communicating with neighboring nodes. In the past, sensors are connected by wire lines. Today, this

[*]This work is co-sponsored by the Lee and MTI Center for Networking Research at the National Chiao Tung University and the MOE Program for Promoting Academic Excellence of Universities under grant numbers A-91-H-FA07-1-4 and 89-E-FA04-1-4.

environment is combined with the novel *ad hoc* networking technology to facilitate inter-sensor communication [8]. The flexibility of installation and configuration is greatly improved. A flurry of research activities have recently been commenced in sensor networks.

With sensor networks, the physical world can interact with the internet more closely. Grouping thousands of sensors together may revolutionize information gathering. For example, a disaster detector may be set up so that temperatures of a forest can be monitored by sensors to prevent small harmless brush fires from becoming monstrous infernos. Similar techniques can be applied to flood and typhoon detection. Another application is environment control; sensors can monitor factors such as temperature and humidity and feed these information back to a central air conditioning and ventilation system. By attaching sensors on vehicles, roads, and traffic lights, traffic information can be fed back to the traffic control center immediately. Location-based services can be combined with sensor networks. We can dispatch a mobile agent following a person to provide on-site services (such applications might be attractive for disability people who have such as hearing or visual problems). Sensors may also be used in combination with GPS to improve positioning accuracy. However, many issues remain to be resolved for the success of sensor networks.

- ***Scalability:*** Since a sensor network typically comprises a large number of nodes, how to manage these resources and information is not an easy job. Distributed and localized algorithms are essential in such environments [1, 6, 7]. Also, scalability is a critical issue in handling the related communication problems. In [17, 18, 19], the *coverage* and *exposure* of an irregular sensor network are formulated as computational geometry problems. This coverage problem is related to the Art Gallery Problem and can be solved optimally in a 2D plane, but is shown to be NP-hard in the 3D case [10]. Regular placement of sensors and their sensing ability are discussed in [4] and [13].

- ***Stability:*** Since sensors are likely to be installed in outdoor or even hostile environments, it is reasonable to assume that device failures would be regular and common events. Protocols should be stable and fault-tolerant.

- ***Power-saving:*** Since no plug-in power is available, sensor devices will be operated by battery powers. Energy conservation should be kept in mind in all cases. Energy consumption of communications might be a major factor. Techniques such as data fusion may be necessary [3], but the timeliness of data should be considered too. Data dissemination is investigated in [5]. Mobile agent-based solutions are sometimes more power-efficient [9].

Since sensor networks are typically used to monitor the environment, one fundamental issue is the *location-tracking problem*, whose goal is to trace the roaming paths of moving objects in the network area [15, 20, 11, 16, 14]. This problem is similar to the location-update problem in PCS networks, but is more challenging in two senses: (1) there are no central control mechanism and backbone network in such environment, and (2) the wireless communication bandwidth is very limited. In this paper, we propose a novel protocol based on the *mobile agent* paradigm. Once a new object is detected, a mobile agent will be initiated to track the roaming path of the object. The agent is mobile since it will choose the senor closest to the object to stay. In fact, the agent will follow the object by hopping from sensor to sensor. The agent may invite some nearby slave sensors to cooperatively position the object and inhibit other irrelevant (i.e., farther) sensors from tracking the object. Using mobile agents may have two advantages. First, the sensing, computing, and communication overheads can be greatly reduced.

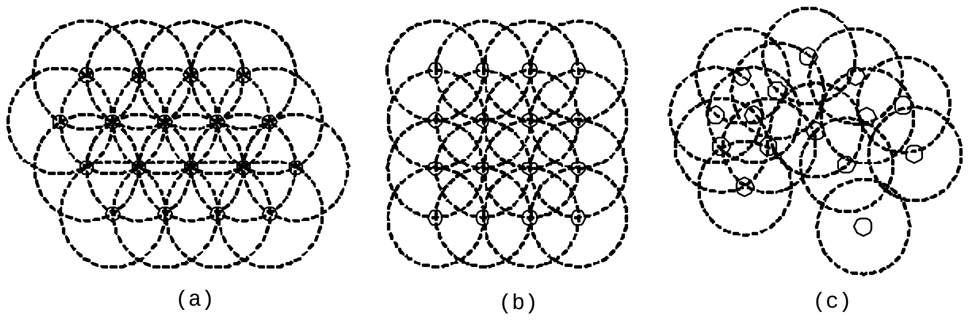

Figure 1: (a) Triangular, (b) square, and (c) irregular sensor networks.

In this work, we will address the delivery and fusion of the tracking results [21]. Second, on-site or follow-me services may be provided by mobile agents. Our prototyping of the location-tracking mobile agent based on IEEE 802.11b NICs and our experimental experiences are also reported.

The organization of this paper is as follows. Section 2 describes our network model and defines the location-tracking problem. Our protocol based on mobile agents is presented in Section 3. Fusion and delivery of tracking history are discussed in Section 4. Our prototyping experiences and some simulation results are given in Section 5. Section 6 draws our conclusions.

## 2 Network Model and Problem Statement

We consider a sensor network, which consists of a set of sensor nodes placed in a 2D plane. Sensors may be arranged as a regular or irregular network, as shown in Fig. 1. However, unless otherwise stated, throughout the discussion we will assume a triangular network as illustrated in Fig. 1(a), our framework should be easily extended to other regular, or even irregular, networks (this will be commented in Section 3-3). In order to track objects' routes, each sensor is aware of its physical location as well as the physical locations of its neighboring sensors. Each sensor has sensing capability as well as computing and communication capabilities, so as to execute protocols and exchange messages.

Each sensor is able to detect the existence of nearby moving objects. We assume that the sensing scope is $r$, which is equal to the side length of the triangles [1]. Within the detectable distance, a sensor is able to determine its distance to an object. This can be achieved either by the fly time or signal strength that are transmitted by the object, or of the signals that are transmitted by the sensor and reflected by the object.

We assume that three sensors are sufficient to determine the location of an object. Specifically, suppose that an object resides within a triangle formed by three neighboring sensors $S_1$, $S_2$, and $S_3$ and that the distances to the object detected by these sensors are $r_1$, $r_2$, and $r_3$, respectively. As shown in Fig. 2(a), by the intersections of the circles centered at $S_1$ and $S_2$, two possible positions of the object can be determined. With the assistance of $S_3$, the precise position can be determined. (It should be noted that in practice errors may exist, and thus more sensors will be needed to to improve the accuracy.)

The goal of this work is to determine the roaming path of a moving object in the sensor network. The trace of the object should be reported to a location server from time to time, depending on whether

[1]In practice, $r$ should be slightly larger than the side length. We make such an assumption for ease of presentation.

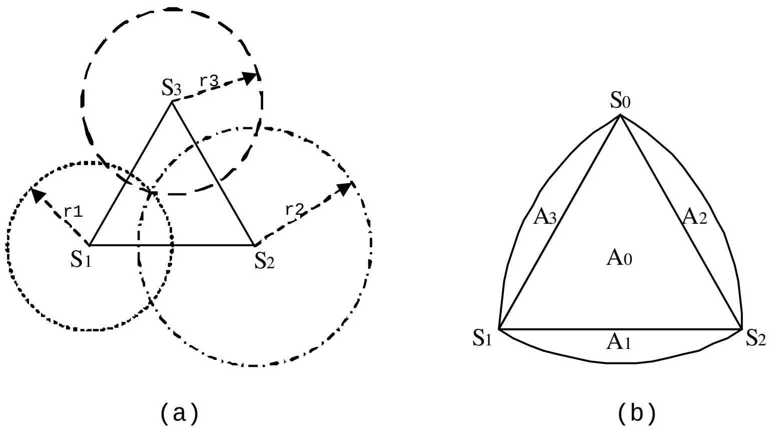

Figure 2: (a) Positioning example and (b) working area and backup areas.

this is a real-time application or not. The intersection of the sensing scopes of three neighboring sensors is as shown in Fig. 2(b). We further divide the area into one *working area* $A_0$ and three *backup areas* $A_1, A_2$, and $A_3$. Intuitively, the working area defines the scope where these three sensors work normally, while the backup areas specify when "handover" should be taken.

## 3 The Location Tracking Protocol

### 3.1 Basic Idea

Our location-tracking protocol is derived by the cooperation of sensors. Whenever an object is detected, an *election process* will be conducted by some nearby sensors to choose a sensor, on which an agent will be initiated, to monitor the roaming behavior of the object. As the object moves, the agent may migrate to a sensor that is closer to the object to keep on monitoring the object. Fig. 3 illustrates this concept, where the dash line is the roaming path of the object, and arrows are the migration path of the agent. By so doing, the computation and communication overheads can be reduced significantly.

Recall that positioning an object requires the cooperation of at least three sensors. The mobile agent, called the *master*, will invite two neighboring sensors to participate by dispatching a *slave* agent to each of them. These three agents (master and slaves) will cooperate to perform the trilateration algorithm [1]. From time to time, the slaves will report their sensing results to the master agent, who will then calculate the object's precise locations. As the object moves, these slave agents may be revoked and reassigned. Certain signal strength thresholds will be used to determine when to revoke/reassign a slave agent. The details will be given later. In Fig. 3, those sensors that ever host a slave agent are marked by black. We comment that although our development is based on the cooperation of two slave agents, it will be straightforward to extend our work to more slave agents to improve the positioning accuracy. To reduce the amount of data to be carried on, a master may decide to forward some tracking histories to the location server. This issue will be further addressed in Section 4.

We now discuss how slave agents are revoked and reassigned. Observe the top part of Fig. 3.

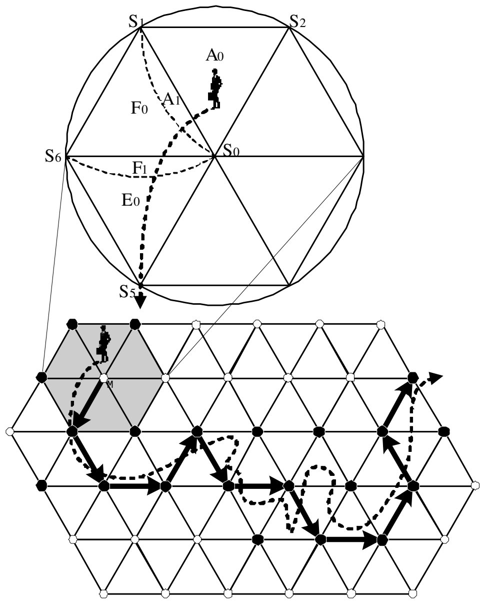

Figure 3: Roaming path of an object (dash line) and the migration path of the corresponding master agent (arrows). Sensors that ever host a slave agent are marked by black.

When resident in the working area $A_0$, the object is tracked by sensors $S_0$, $S_1$, and $S_2$. On entering the backup area $A_1$, since the signals received by $S_2$ will reduce to a level below a threshold, the slave agent at $S_2$ will be revoked and a new slave will be issued to $S_6$. Similarly, on entering the backup area $F_1$, the slave at $S_1$ will be revoked, and a new one will be issued to $S_5$. As the object passes $S_5$, the master itself will lose the target, in which case the master will migrate itself to $S_5$. All old slaves will be revoked and new slaves will be invited.

When an object is in the backup areas of some sensors, it is possible that it can be sensed by more than three sensors. To reduce the sensing overheads, master and slave agents can inhibit other irrelevant sensors from monitoring the object. This concept is illustrated in Fig. 4. The object is currently in area $A_0$. Sensors $S_3, S_4, \ldots, S_{11}$, which may sometimes detect the object, will be inhibited from tracking this object by warning signals that are issued periodically by the agents in $S_0, S_1$, and $S_2$.

## 3.2 Protocol Details

Below, we formally develop our tracking protocol. Since there may exist multiple objects in the network, we have to assume that sensors can distinguish one object from the other. This can be done by having each object periodically send a unique ID code. Otherwise, some mechanism is needed for sensors to combine proper signals from proper sensors to differentiate objects.

We consider an environment with multiple objects. However, since the processing of each individual object is independent, the following discussion will focus on only one particular object. For each object, three or even more sensors will be able to detect its existence. Fig. 5 shows the state transition diagram of each sensor. (It should be noted that for different objects, a sensor may stay in different states.) Initially, each sensor is in the *idle* state and performs the *Basic Protocol*. Under this state, a sensor will continuously detect any object appearing in its sensing scope. Once detecting a

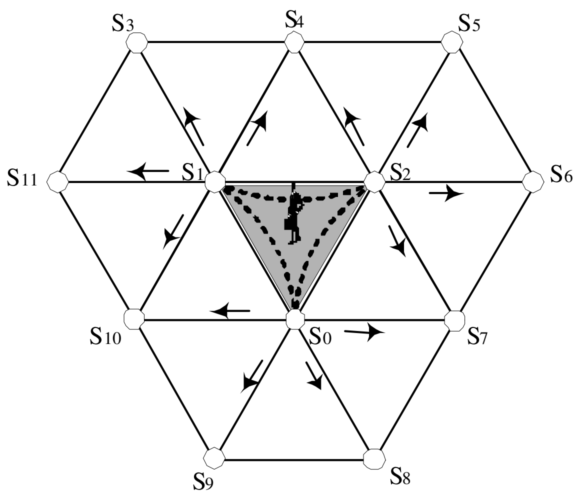

Figure 4: Inhibiting farther sensors $S_3, S_4, \ldots, S_{11}$ from monitoring the object.

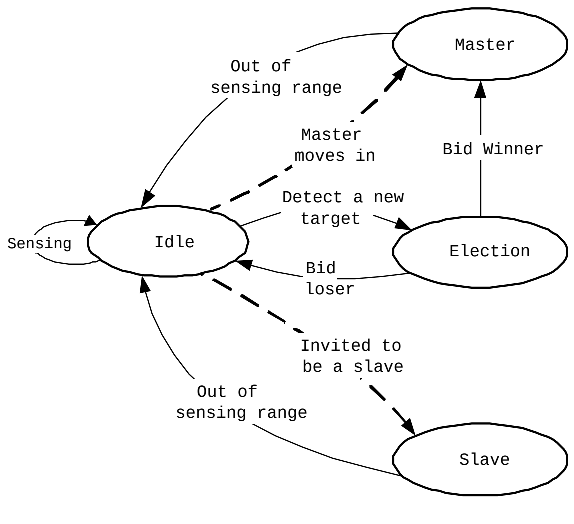

Figure 5: State transition diagram of a sensor (for one particular object).

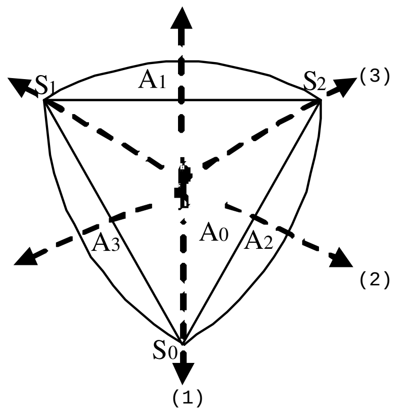

Figure 6: Possible roaming tracks for an object to leave a triangle.

new object, the sensor will enter the *election* state and perform the *Election Protocol* to bid for serving as a master. Most likely, the sensor that is closest to the object will win and become the master agent, which will then dispatch two slave agents to two nearby sensors. The master will go to the *master* state and perform the *Master Protocol*, while the slaves will go to the *slave* state and perform the *Slave Protocol*. To prevent too frequent moves of the agents, as long as the object remains in the working area, the states will not change. However, once the object enters the backup areas, the roles of master and slave may be changed. In this case, an idle sensor may be invited to serve as a master or slave. Another case that a sensor may stay in the idle state is when it detects an object in its backup areas and keeps on receiving inhibiting messages from neighboring sensors. This is reflected by the self-looped transition for the idle state.

Fig. 6 shows six tracks that an object may leave a triangle. Suppose that the master is currently in $S_0$, and the two slaves are in $S_1$ and $S_2$. By symmetry, these can be reduced 3 tracks (numbered by 1 to 3). For track 1, the master discovers two slaves losing the target simultaneously. So the master will revoke all slaves and invite two new slaves. For track 2, only the slave agent in $S_1$ will be revoked, and a new one will be invited. For track 3, the master discovers one slave as well as itself losing the target. In this case, the master should migrate itself to the sensor that can still detect the object (typically with the strongest receive signals) and revoke all current slaves. After moving to the new sensor, two new slaves should be invited. Finally, we comment that the object may move too fast to be detected. If so, sensors may suddenly lose the target. As a last resort, all agents,when losing the object for a timeout period, will be dissolved. Since no inhibiting message will be heard, all sensors must remain in the idle state for this particular object, and new election process will take place to choose a new master to track this object. Our protocol is thus quite fault-tolerant in this sense.

Each sensor will keep an *object list (OL)* to record the status of all targets in its sensing scope. Each entry in OL is indexed by the object's unique identity, denoted by ID. For each object, there are two sub-fields: *status* and *time-stamp*. *ID.status* can be one of the four values: *Master*, *Slave*, *Standby*, and *Inhibited*. *ID.time-stamp* is the time when the record is last updated.

Seven types of control messages may be sent by our protocol.

(1) *bid_master(ID, sig):* This is for a sensor to compete as a master for object ID, if no inhibiting record has been created in OL for ID. The parameter $sig$ reflects the receive signal strength for

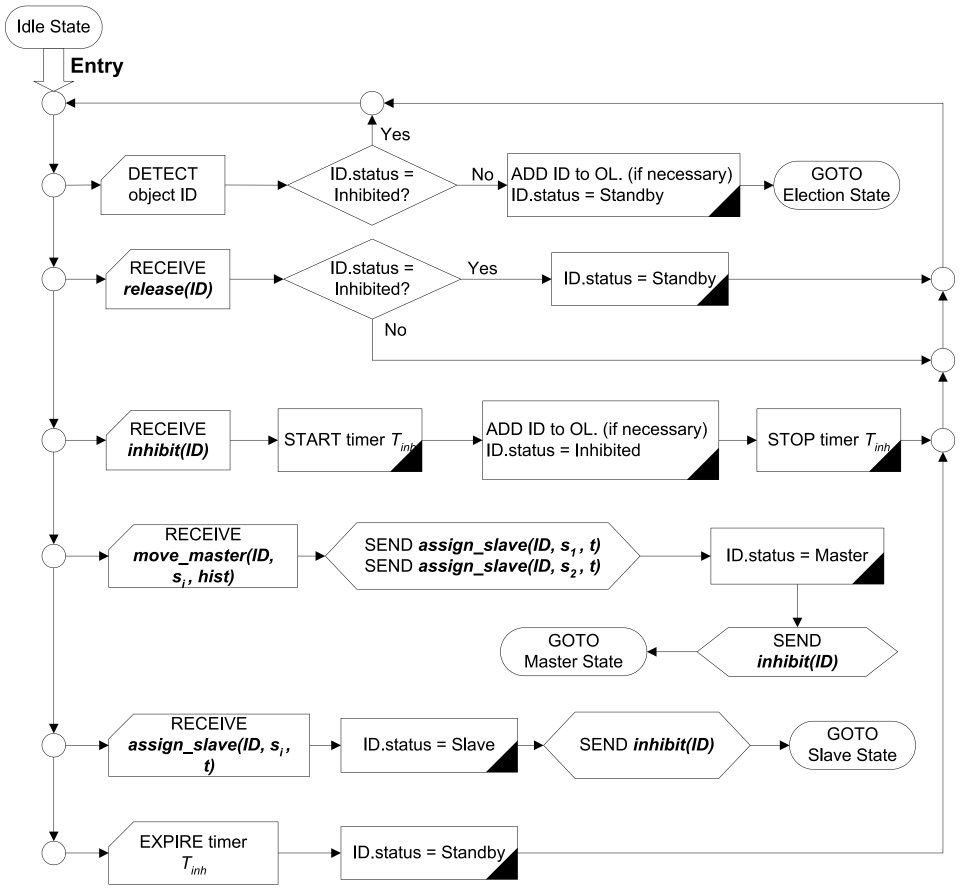

Figure 7: The Basic Protocol.

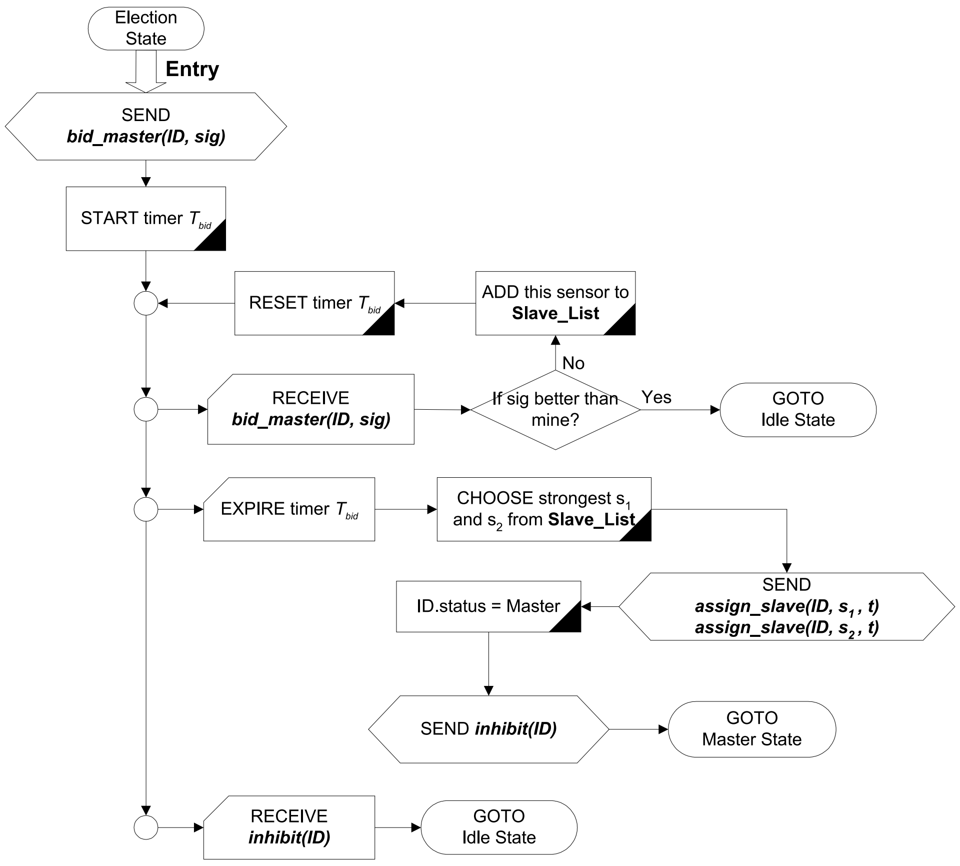

Figure 8: The Election Protocol.

this object.

(2) *assign_slave(ID, $s_i$, t):* This is for a master to invite a nearby sensor $s_i$ to serve as slave agent for object ID for an effective time interval of $t$.

(3) *revoke_slave($s_i$):* This is for a master to revoke its slave at sensor $s_i$.

(4) *inhibit(ID):* This is a broadcast message for a master/slave to inhibit neighboring irrelevant sensors from tracking object ID. The effective time of the inhibiting message is defined by a system parameter $T_{inh}$.

(5) *release(ID):* This is to invalidate an earlier inhibiting message.

(6) *move_master(ID, $s_i$, $hist$):* A master uses this message to migrate itself from its current sensor to a nearby sensor $s_i$, where $hist$ carries all relevant codes/data/roaming histories related to object ID.

(7) *data(ID, $sig$, $ts$):* A slave uses this packet to report to its master the tracking results ($sig$ =signal strength and $ts$ = timestamp ) for ID.

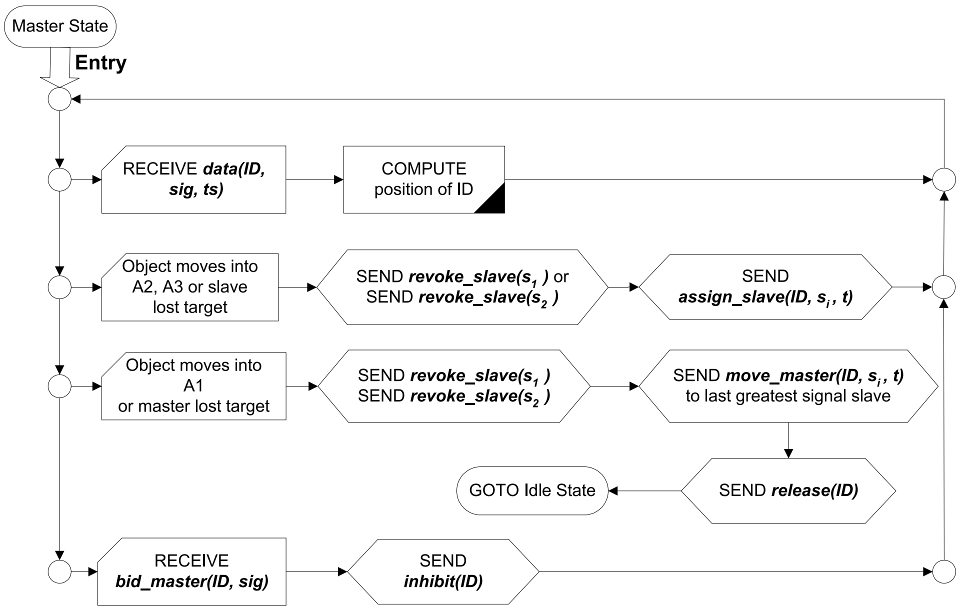
Master State
Entry
RECEIVE data(ID, sig, ts)
COMPUTE position of ID
Object moves into A2, A3 or slave lost target
SEND revoke_slave(s1 ) or SEND revoke_slave(s2 )
SEND assign_slave(ID, si, t)
Object moves into A1 or master lost target
SEND revoke_slave(s1 ) SEND revoke_slave(s2 )
SEND move_master(ID, si, t) to last greatest signal slave
GOTO Idle State
SEND release(ID)
RECEIVE bid_master(ID, sig)
SEND inhibit(ID)

Figure 9: The Master Protocol.

Below, we formally present our four protocols. The Basic Protocol is shown in Fig. 7. This is an endless loop containing six event-driven actions. The first one describes the reaction when detecting an object. If an inhibiting record exists, it will ignore the object. Otherwise, the sensor will go to the election state. The next four events describe the reactions when receiving a message from a neighboring sensor. In particular, if an *inhibit(ID)* message is received, a timer $T_{inh}(ID)$ will be set. The last event describes the reaction when the above timer expires, in which case the object's status will be changed to Standby and the sensor will be allowed to monitor this object.

The Election Protocol is shown in Fig. 8. In the beginning, a *bid_master* message will be sent and a timer $T_{bid}(ID)$ will be set. Then the sensor will wait for three possible events to occur: receiving *bid_master*, receiving *inhibit*, and finding timer $T_{bid}$ expired. Signal strength will be used in the competition. Depending of different events, the sensor will go to the Master or Idle state.

Fig. 9 shows the Master Protocol. The first event is to collect data from neighboring sensors. The next two events are for slave agents and the master agent when losing the target, respectively. Note that the areas A1, A2, and A3 refer to Fig. 2(b). The last event is to inhibit irrelevant sensors from monitoring the object.

The Slave Protocol is shown in Fig. 10. The first event controls the timing, by timer $T_{rep}$, to report data to the master. The second event is for the master to revoke the slave. The last event is to inhibit other irrelevant sensors.

## 3.3 Extension to Irregular Network Topologies

The above discussion has assumed a triangular sensor network topology. In the following, we briefly discuss how to extend our work to handle irregular deployment of sensors.

The election process does not need to be changed because sensors can still bid for serving as a master/slave based on their receive signal strengths. However, the rules to migrate masters/slaves

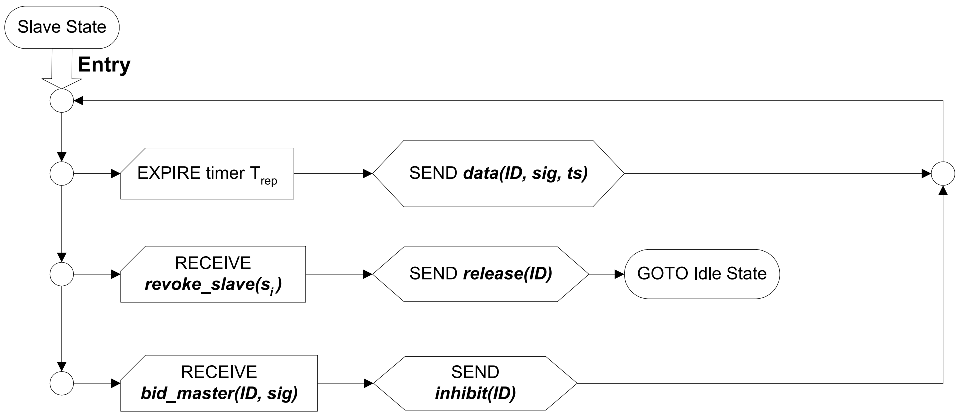

Figure 10: The Slave Protocol.

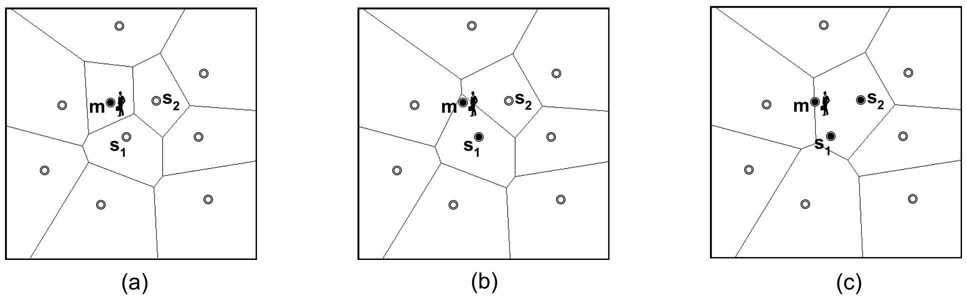

Figure 11: Using Voronoi graphs to find the master and slaves: (a) the Voronoi graph of all vertics, (b) the Voronoi graph after removing the master, and (c) the Voronoi graph after removing the master and first slave.

need to be modified slightly as follows. Sensors need to know the locations of at least their two-hop neighbors. The working and backup areas are redefined based on the sensing scope, $r$, of each sensor. Specifically, there is a predefined value $r' < r$. The working area of a sensor is the circle centered at itself with radius $r'$. The rest of the area is the backup area. As before, we still use one master and two slaves to track an object (although more slaves may be used). Whenever the master finds the object moves into the backup area of itself or any of the slaves, the corresponding agent will be revoked and new agent will be assigned.

One interesting theoretical problem is how to define the master and two slaves given an object in an irregular network. This can be related to the classical *Voronoi* graph problem in geometry [2]. Given a set of points $V$ in a 2D plane, the Voronoi graph partitions the plane into $|V|$ segments such that each segment contains all points that is closest to the (only) vertex in the segment. As a result, if $V$ is the set of all sensors, the sensor of the segment containing the object will serve as its master agent. Fig. 11(a) shows an example. The problem can be solved by a divide-and-conquer solution in time complexity $O(|V| \log |V|)$ [2].

The next two sensors that are closest to the object will serve as the slave agents. This can be found recursively as follows. Specifically, let $m$ be the master sensor. We can construct the Voronoi graph again based on the vertex set $V - \{m\}$. Then the sensor, say $s_1$, of the segment containing the object will serve as the first slave. For example, Fig. 11(b) is the new Voronoi graph after removing the master sensor $m$. Similarly, to find the second slave, we repeat the process by constructing the Voronoi graph of $V - \{m, s_1\}$. Then the sensor, say $s_2$, of the segment containing the object will serve as the second slave. An example is in Fig. 11(c).

The advantage of using the Voronoi graph is as follows. For a particular location of the object, we can sort its distance to each sensor and pick the first three sensors closest to it. The complexity is $O(|V| \log |V|)$. However, whenever the object moves, the list needs to be re-sorted. The computational cost increases as time proceeds. If the above approach is used, we only need to pre-compute $1 + \binom{|V-1|}{1} + \binom{|V-1|}{2}$ Voronoi graphs. So the saving of using Voronoi graphs is clear when we need to track the object for longer time.

## 4 Fusion and Delivery of Tracking Results

One issue not yet addressed is when a master agent should deliver its tracking result to the outside world. We assume that one of the sensors in the network serves as the gateway connecting to a location server in the wireline network. From time to time, the tracking result should be sent to the location server. We assume that more tracking result will be accumulated as time proceeds. So an optimization problem is that the master agent needs to decide whether it should carry the tracking result from sensor to sensor, or forward the result to the gateway.

We assume that for each object being tracked, the tracking results are generated at a constant rate $r$, and each tracking result is of size $d$ bytes. That is, in time interval $\Delta t$, the amount of tracking result is $\Delta t \cdot r \cdot d$. Further, a sequence of tracking results can be combined with a *fusion factor* $\rho$, $0 \leq \rho < 1$, at a basic cost of $b$ bytes. Specifically, the above tracking results can be compressed into $b + \Delta t \cdot r \cdot d \cdot \rho$ bytes. In most cases, data fusion is beneficial. This is normally happens when data has certain level of dependence. In the following, we propose three data delivery solutions. Note that the first one is in fact not an agent-based solution. It only serves as a referential strategy so as to make comparison to our agent-based solutions.

The first one is called the *Non-Agent-Based (NAB)* strategy. Each sensor works independently and forwards its sensing results back to the gateway from time to time. Note that the sensing result is raw data and needs to be combined with other sensors' sensing results at the gateway to calculate the object's locations. The shortest paths, which are assumed to be supported by the underlying routing protocol, are always used for data delivery. Also, we assume an ideal situation that only the three sensors nearest to the object will track the object.

The second solution is called the *Threshold-Based (TB)* strategy. A predefined threshold value $T$ is given. The master agent will accumulate the tracking results and "carry" the result with it as long as the amount of result does not exceed $T$. Whenever the amount of results (after fusion) reaches $T$, it will be forwarded to the gateway through a shortest path.

The third solution is called the *Distance-Based (DB)* strategy. The delivery action may be taken only when the master agent moves. Basically, the distances from the agent's current and next sensors to the gateway are considered. Suppose that the master agent is currently at sensor $S_i$ and is going to be migrated to sensor $S_{i+1}$. Let $N_i$ be the current amount of tracking results accumulated by the agent before it leaves $S_i$. Also, we assume that $N_{i+1}$ is the expected amount of tracking results that shall be accumulated by the agent at $S_{i+1}$ (this value can be formulated by a constant $\overline{T} \cdot r \cdot d \cdot \rho$, where $\overline{T}$ is the expected residential time of an agent at a sensor).

If the master decides to carry the tracking result with it, the expected cost is:

$$C_1 = N_i + (N_i + N_{i+1}) \times d(g, S_{i+1}),$$

where the first term is the cost to migrate the current result to the next sensor, and the second term is the expected cost to deliver the fused result at the next sensor to the gateway, $g$. Function $d()$ specifies the minimum number of hops between two sensors. If the master decides to deliver its current tracking result to the gateway, the expected cost is:

$$C_2 = N_i \times d(g, S_i) + (b + N_{i+1}) \times d(g, S_{i+1}).$$

Subtracting these two factors, we have

$$C_2 - C_1 = b \times d(g, S_{i+1}) + N_i \times (d(g, S_i) - d(g, S_{i+1})) - N_i.$$

So the master agent will carry the results with it iff $C_1 < C_2$; otherwise, the results will be sent back to the gateway. Since sensors $S_i$ and $S_{i+1}$ are neighbors, $d(g, S_t) - d(g, S_{t+1}) = -1$, 0, or 1. Considering whether the agent is moving away from or closer to the gateway, we simplify the condition into three cases.

- *Move away:* That is, $d(g, S_i) - d(g, S_{i+1}) = -1$. Then we have

$$\begin{aligned} C_1 < C_2 &\equiv d(g, S_{i+1}) > \frac{2N_i}{b} \\ &\equiv d(g, S_i) > \frac{2N_i}{b} - 1. \end{aligned} \tag{1}$$

- *Move parallel:* That is, $d(g, S_i) - d(g, S_{i+1}) = 0$. Then we have

$$\begin{aligned} C_1 < C_2 &\equiv d(g, S_{i+1}) > \frac{N_i}{b} \\ &\equiv d(g, S_i) > \frac{N_i}{b}. \end{aligned} \tag{2}$$

- *Move closer:* That is, $d(g, S_i) - d(g, S_{i+1}) = 1$. Then the agent will always carry the data with it because

$$C_1 < C_2 \equiv b \times d(g, S_{i+1}) > 0 \equiv TRUE. \tag{3}$$

## 5 Prototyping Experiences and Simulation Results

### 5.1 Prototyping Experiences

In order to verify the feasibility of the proposed protocol, we have prototyped a system based on IEEE 802.11b NICs. Signal strength is used as the criterion to position objects. Specifically, one laptop equipped with a Lucent ORiNOCO 802.11b WaveLAN card is used to simulate an object. A number of laptops, also equipped with IEEE 802.11b cards, are placed in triangular/square patterns to emulate sensor nodes, as shown in Fig. 12. The object can roam around and will measure beacon

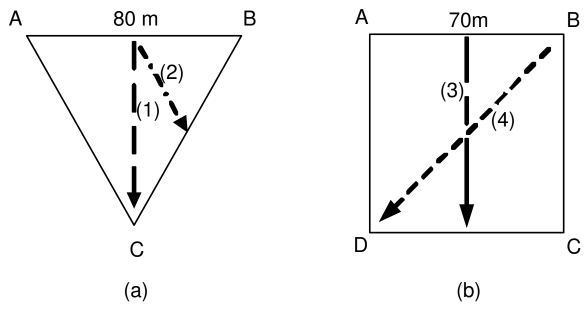

Figure 12: Experimental environment: (a) triangular sensor network and (b) square sensor network. Dash lines represented tested roaming paths.

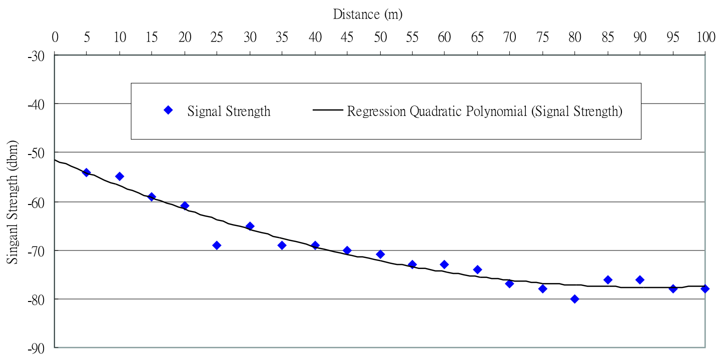

Figure 13: Experiment of signal strength vs. distance for IEEE 802.11b.

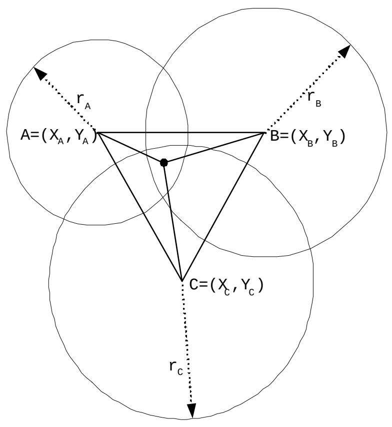

Figure 14: The position approximation algorithm.

strengths transmitted from different sensors. For better accuracy, we average ten samples in one second.

First, we measure the degradation of signal strength versus distance. Fig. 13 shows one set of data that we collected. For every 5 meters from 0 to 100 meters a measurement is recorded. As can be expected, signal strengths received from IEEE 802.11b are not very stable. We use the “regression quadratic polynomial” to smooth out the curve, as illustrated by the solid line in Fig. 13. The curve is used to convert a received signal strength to an estimated distance.

Since signal strength is not an accurate measurement, the aforementioned trilateration algorithm can not be applied directly. In fact, as one may expect, signal strengths change all the time, even under a motionless situation. Certain gaps inherently exist between estimated distances and actual distances. The real situation is as shown in Fig. 14, where the three estimated circles centered at sensors have no common intersection.

To solve the problem, we propose an approximation algorithm as follows. Let $A$, $B$, and $C$ be the sensor nodes, which are located at $(x_A, y_A)$, $(x_B, y_B)$, and $(x_C, y_C)$, respectively. For any point $(x, y)$ on the plane, we then define a difference function

$$\begin{aligned}\sigma_{x,y} = {} & |\sqrt{(x-x_A)^2+(y-y_A)^2} - r_A| \\ & + |\sqrt{(x-x_B)^2+(y-y_B)^2} - r_B| \\ & + |\sqrt{(x-x_C)^2+(y-y_C)^2} - r_C|,\end{aligned}$$

where $r_A$, $r_B$, and $r_C$ are the estimated distances to A, B, and C respectively. The location of the object is determined to be the point $(x, y)$ among all points such that its difference function $\sigma_{x,y}$ is minimized. In our experiment, we consider only integer grid points on the plane. We measure the location of the object every second. Furthermore, to take sudden fluctuation of signal strength into account, we enforce a condition that the object does not move faster than 5 meters per second. As a result, when searching for the object's location, only those points in $(x \pm 5, y \pm 5)$ are evaluated for their difference functions, where $(x, y)$ represents the location in the previous measurement.

Our experiments were done in an outdoor, plain area with no obstacles. Two roaming paths as illustrated in Fig. 12(a) were tested. For roaming path (1), three sets of results are shown in Fig. 15.

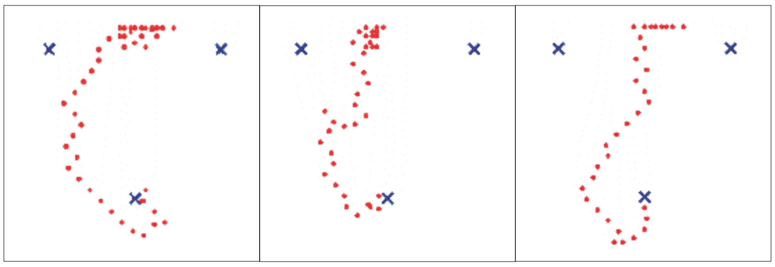

Figure 15: Tracking result of path (1) in Fig. 12(a).

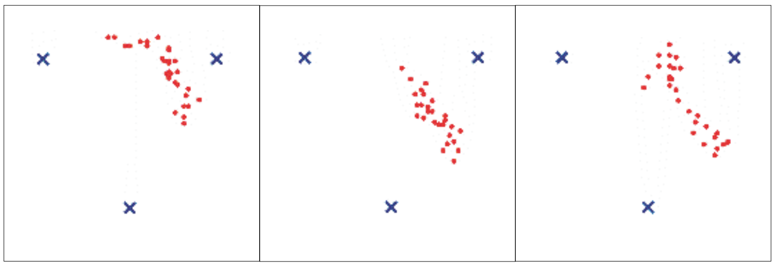

Figure 16: Tracking result of path (2) in Fig. 12(a).

For roaming path (2), the results are demonstrated in Fig. 16. As can be seen, the predicted paths are close to the actual roaming paths, but there are still large gaps yet remaining to be improved further.

We have also tested the arrangement in Fig. 12(b), where four sensors arranged as a square are used. The extension for the tracking protocol and positioning algorithm is straightforward. Our tested results are shown in Fig. 17 and Fig. 18.

A larger-scale experiment with 12 sensors is shown in the Fig. 19(a). With our agent-based approach, the object is tracked by the four sensors with the strongest signals. The other distanced sensors will be inhibited from monitoring the object (and thus reporting their tracking results). On the contrary, if all sensors which can detect the object are allowed to track the object, the tracking result will be as shown in Fig. 19(b). Surprisingly, the result shows that the positioning accuracy only improves very slightly. We believe that this is because the signal strength from a distanced sensor is typically very unstable. This usually enlarges the range of error. As a result, using our agent-based approach not only reduces the amount of data being transmitted, but also remains the same level of

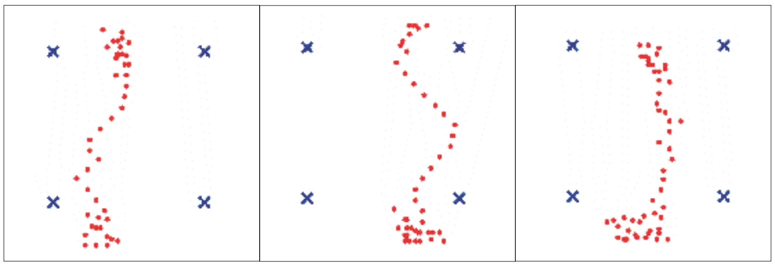

Figure 17: Tracking result of path (3) in Fig. 12(b).

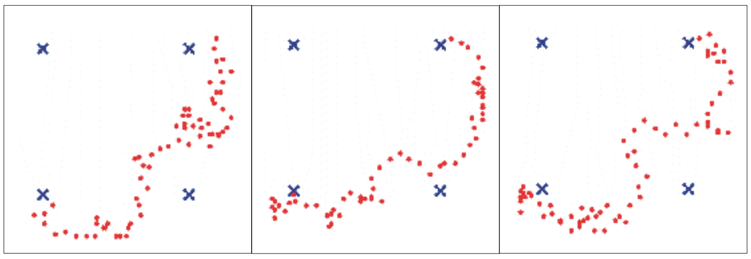

Figure 18: Tracking result of path (4) in Fig. 12(b).

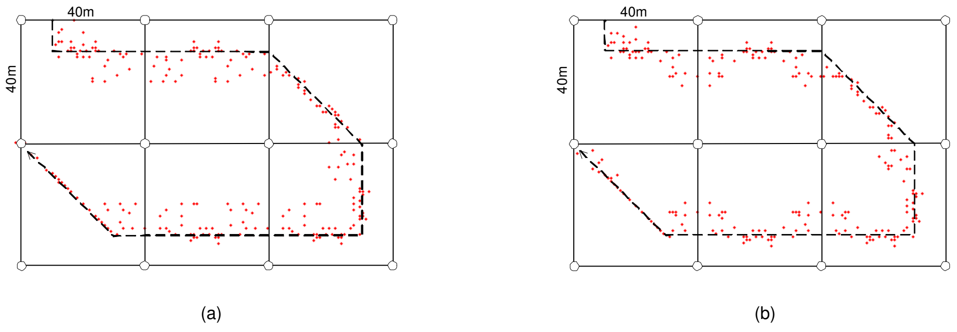

Figure 19: Comparison of tracking accuracy: (a) agent-based approach by using at most 4 sensors and (b) non-agent-based approach. Dashed lines are the real roaming paths, and dots are the tracking results.

positioning accuracy.

## 5.2 Simulation Results

To verify the advantage of using our agent-based approach, we have developed a simulator. Sensors are deployed in a 10,000m x 10,000m environment with triangular topology. The distance between two neighboring sensors is 80m. The gayeway is located at the center of the network. Each control packet is 2 bytes. Each location is represented by 2 bytes. The IP routing header is assumed, wtih each header equal to 2 bytes and MTU as large as 500 bytes.

The Random Way Point Model [12] is used to simulate the mobility of objects. The initial locations of objects are chosen randomly. Each object alternates between moving and pausing states. On entering the moving state, the object's next destination is randomly chosen from $(x \pm 15, y \pm 15)$, where $(x, y)$ is its current location. Note that locations outside the boundary are not considered. Under moving state, the object moves at a constant speed of an uniform distribution between 1$\sim$3 m/sec. After arriving at its destination, the object will pause a period with an exponential distribution of mean = 5 sec.

We first experiment on different threshold values of $T$ for the *TB* strategy. The result is in Fig. 20(a). We measure the average traffic load. A $T$ significantly less than the largest MTU is not good due to high packet header overheads. On the contrary, tuning $T$ too large is also inefficient because the master agent will need to carry too much history while traveling. The figure suggests

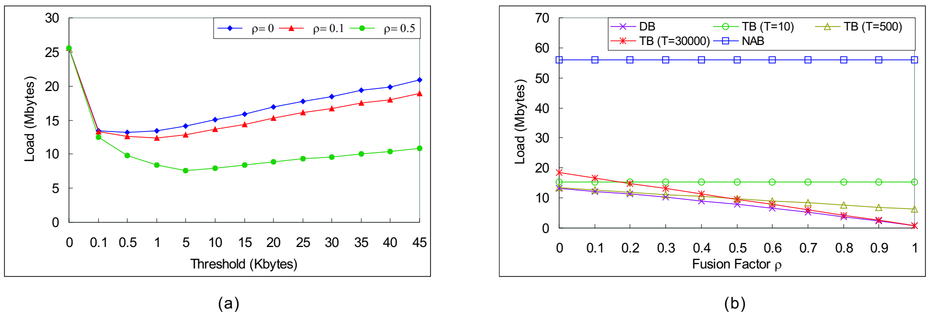

Figure 20: Simulation results: (a) the threshold $T$ of *TB* vs traffic load, and (b) the data fusion factor $\rho$ vs. traffic load.

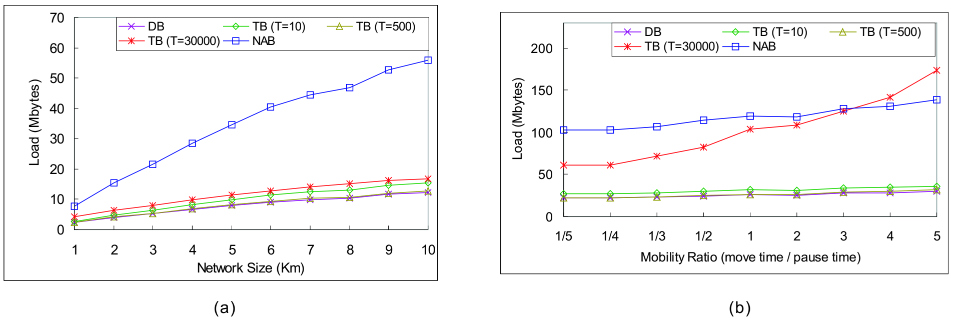

Figure 21: Simulation results: (a) network size vs. traffic load ($\rho = 0.1$), and (b) mobility ratio vs. traffic load ($\rho = 0.1$).

that a $T$ equal to or slightly larger than the largest MTU would be a good choice. Fig. 20(b) further demonstrates the effect of the fusion factor $\rho$. We compare different strategies. The *DB* strategy performs the best. The *TB* also performs very well, if proper $T$ can be selected. In all cases, *NAB* performs the worst.

In the Fig. 21(a), we change the network size to visualize the effect. It is reasonable that larger networks incur higher traffics due to longer delivery paths. This justifies the importance of using our agent-based strategies. In Fig. 21(b), we further vary the mobility ratio, which is defined to be the ratio of moving time to pausing time. A higher mobility ratio indicates more frequent change of master agents. DB and TB with lower thresholds are less sensitive to mobility. With a too large threshold, TB will degrade significantly because the overhead for agents to carry tracking results would be significant as the mobility ratio increases.

To summarize, we conclude that *DB* performs well in all cases. *TB* is quite simple, but one needs to be cautious in choosing its threshold. These strategies outperform NAB by 60$\sim$80% in terms of average traffic load.

## 6 Conclusions

We have proposed a novel location-tracking protocol for regular and irregular sensor networks. A mobile-agent approach is adopted, which enables agents to roam around to follow the moving objects, hence significantly reducing the communication and sensing overheads. A data fusion model is proposed, and several data delivery strategies are proposed and evaluated. We have prototyped a system based on the idea using IEEE 802.11b NICs, where signal strengths are used as the criterion to measure objects' positions. While the prototyping is proved to work correctly, the accuracy still has rooms to be improved further.

## References

[1] A. Savvides, C.C. Han, M.B. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Procs. of MobiCOM*, 2001.

[2] F. Aurenhammer. Voronoi diagrams: a survey of a fundamental geometric data structure. pages 345–405, 1991.

[3] B. Horling, R. Vincent, R. Mailler, J. Shen, R. Becker, K. Rawlins, and V. Lesser. Distributed Sensor Network for Real Time Tracking. In *Proc. of the 5th international conference on Autonomous agents*, pages 417–424, 2001.

[4] B. Huang, W. Zhang, and Z. Guo. A study of spatial structures of sensor networks and multi-agent negotiation strategies, (this is an appendix to a quarterly report.), 2001. http://www.cs.wustl.edu/ zhang/projects/dcmp/.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. of MobiCOM*, pages 56–67, 2000.

[6] C. Savarese, J. Rabaey, J. Beutel. Locationing in distributed ad-hoc wireless sensor networks. In *Proc. of the ICASSP*, 2001.

[7] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. In *Proc. of MobiCOM*, pages 263–270, 1999.

[8] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[9] Hairong Qi, S.S. Iyengar, and K. Chakrabarty. Multiresolution data integration using mobile agents in distributed sensor networks. *IEEE Tran. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 31(3):383–391, 2001.

[10] J. O'Rourke. Computational geometry column 15. *International Journal of Computational Geometry and Applications*, 2(2):215–217, 1992.

[11] J. Rabaey, J. Ammer, J.L. da Silva Jr., D. Patel. Picoradio: Ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes. In *VLSI, 2000. Proceedings. IEEE Computer Society Workshop*, pages 9–12, 2000.

[12] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[13] K. Chakrabarty, S.S. Iyengar, Hairong Qi, and Eungchun Cho. Coding theory framework for target location in distributed sensor networks. In *Proc. Intl. Symposium on Information Technology: Coding and Computing*, pages 130–134, 2001.

[14] Nirupama Bulusu, John Heidemann, Deborah Estrin. Gps-less low-cost outdoor localization for very small devices. In *IEEE Personal Communications*, pages 28–34, 2000.

[15] P. Enge, and P. Misra. Special issue on gps: The global positioning system. In *Proc. of the IEEE*, pages 3–15, 1999.

[16] Paramvir Bahl, and Venkata N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *INFOCOM*, pages 775–784, 2000.

[17] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless Ad-Hoc sensor networks. In *Proc. of MobiCOM*, pages 139–150, 2001.

[18] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage Problems in Wireless Ad-hoc Sensor Networks. In *Proc. of INFOCOM*, pages 1380–1387, 2001.

[19] S. Meguerdichian, S. Slijepcevic, V. Karayan and M. Potkonjak. Localized algorithms in wireless ad-hoc networks: location discovery and sensor exposure. In *Proc. of MobiHOC*, pages 106–116, 2001.

[20] van Diggelen, F. Indoor gps theory & implementation. In *Position Location and Navigation Symposium, 2002 IEEE*, pages 240–247, 2002.

[21] K. Whitehouse and D. Culler. Calibration as parameter estimation in sensor networks. In *Proceedings of the first ACM international workshop on Wireless sensor networks and applications*, pages 59–67, 2002.