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

Email: {yctseng, spkuo, leehw, cfhuang}@csie.nctu.edu.tw

The wireless sensor network is an emerging technology that may greatly aid humans by providing ubiquitous sensing, computing and communication capabilities, through which people can more closely interact with the environment wherever they go. 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 is neither a central control mechanism nor a backbone network in such an 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.

Received 16 June 2003; revised 5 November 2003

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 were connected by wire lines. Today, this environment is combined with the novel *ad hoc* networking technology to facilitate inter-sensor communication [1]. Flexibility in installation and configuration is greatly improved. There has been a flurry of research activity recently 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 could be set up so that temperatures in 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 this 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 that follows a person and provides on-site services (such applications might be attractive for disabled people who have 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, the management of these resources and information is not an easy job. Distributed and localized algorithms are essential in such environments [2, 3, 4]. Also, scalability is a critical issue in handling the related communication problems. In [5, 6, 7], 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 two-dimensional (2D) plane, but is shown to be NP-hard in the 3D case [8]. Regular placement of sensors and their sensing ability are discussed in [9] and [10].
- *Stability.* Since sensors are likely to be installed in outdoor or even hostile environments, it is reasonable to assume that device failure will be regular and common. Protocols should, therefore, be stable and fault-tolerant.
- *Power-saving.* Since no plug-in power is available, sensor devices will be operated by battery power. Energy conservation should be kept in mind in all cases

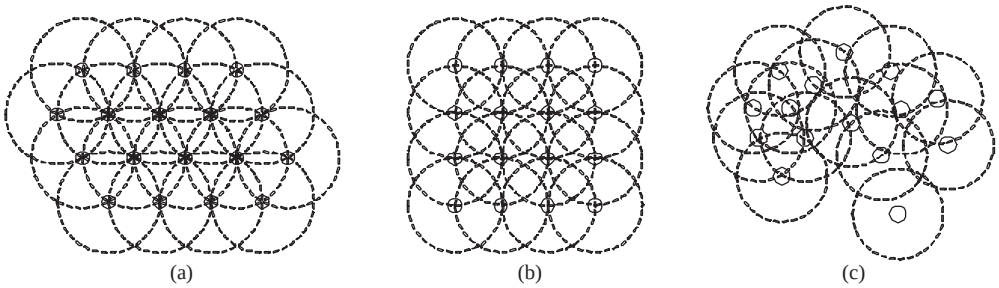

FIGURE 1. (a) Triangular, (b) square and (c) irregular sensor networks.

and energy consumption during communications might be a major factor. Techniques such as data fusion may be necessary [11], but the timeliness of data should be considered too. Data dissemination is investigated in [12]. Mobile agent-based solutions are sometimes more power-efficient [13].

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 [14, 15, 16, 17, 18]. Reference [19] also considers this problem by addressing how to estimate the density or number of targets to be tracked and how to allocate resources to the sensing problem. The location-tracking problem is similar to the location-update problem in PCS networks, but is more challenging in two senses: (1) there is neither a central control mechanism nor a backbone network in such an 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. 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. Second, on-site or follow-me services may be provided by mobile agents. In this work, we will address the fusion of the tracking results. We propose two schemes to send the fused results to the outside world. The first one is called a threshold-based (TB) scheme, which simply forwards the results as the data size reaches an upper bound. The second one is called a distance-based (DB) scheme, which considers both data size and the distance from the mobile agent to the gateway of the network. 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 and Section 6 concludes.

2. NETWORK MODEL AND PROBLEM STATEMENT

We consider a sensor network that 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 Figure 1. However, unless otherwise stated, throughout the discussion we will assume a triangular network as illustrated in Figure 1a; our framework could be easily extended to other regular, or even irregular, networks (this will be discussed in Section 3.3). In order to track objects' routes, each sensor has to be 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 using the fly time or signal strength that are transmitted by the object, or by using 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 Figure 2a, 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 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 this is a real-time application or not. The intersection of the sensing scopes of three neighboring sensors is as shown in Figure 2b. We further divide the area

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

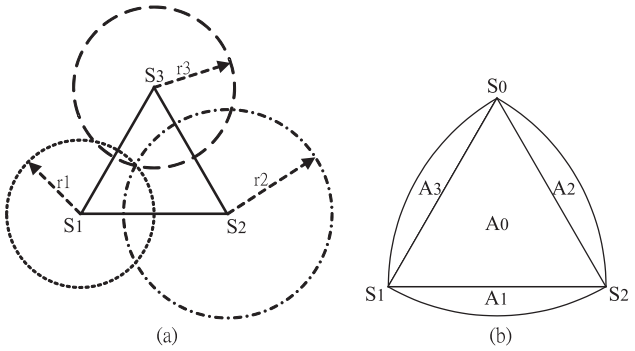

FIGURE 2. (a) Positioning example and (b) working area and backup areas.

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. Figure 3 illustrates this concept, where the dashed line is the roaming path of the object, and arrows are the migration path of the agent. By doing so, 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 [2]. 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 Figure 3, those sensors that host a slave agent are marked in black. We remark 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 Figure 3. 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

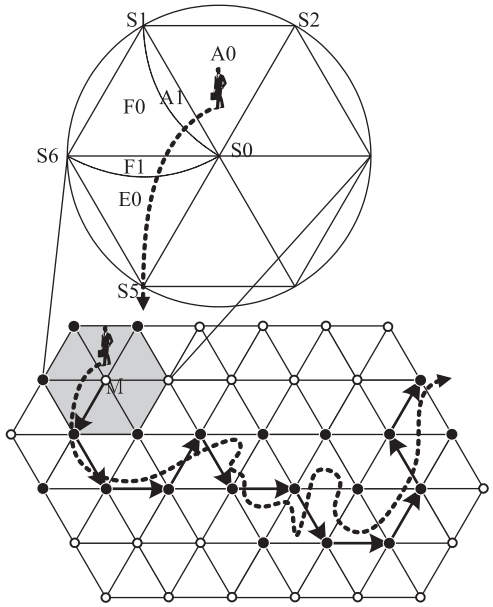

FIGURE 3. Roaming path of an object (dashed line) and the migration path of the corresponding master agent (arrows). Sensors that host a slave agent are marked in black.

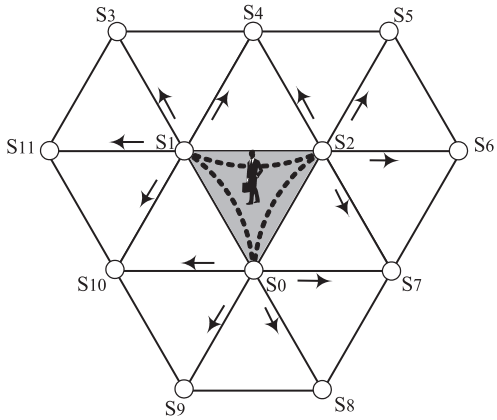

FIGURE 4. Inhibiting farther sensors $S_3, S_4, \ldots, S_{11}$ from monitoring the object.

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 itself will migrate 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 Figure 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

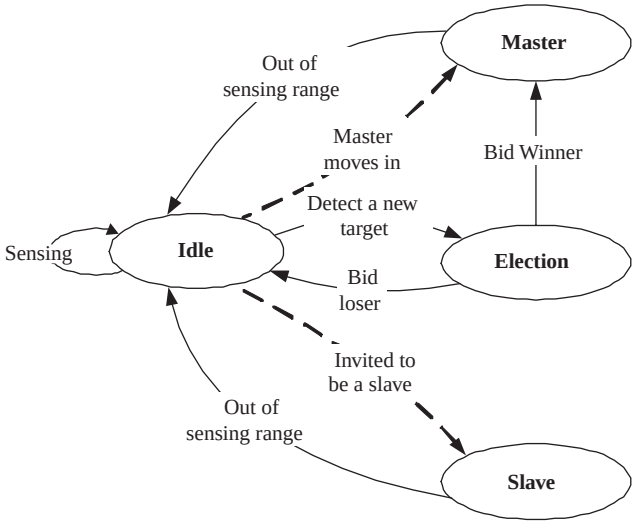

FIGURE 5. State transition diagram of a sensor (for one particular object).

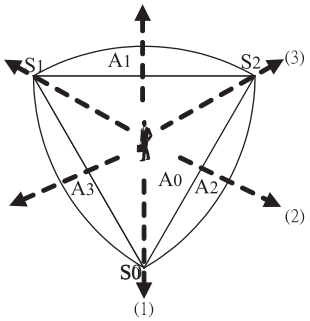

FIGURE 6. Possible roaming tracks for an object to leave a triangle.

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 more sensors will be able to detect its existence. Figure 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 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 agents from moving too frequently, 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 situation in which 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.

Figure 6 shows six tracks along which 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 to three tracks (numbered by 1–3). 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 itself should migrate 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 would like to remark that the object may move too fast to be detected. If so, sensors may suddenly lose the target. In this case, we can let the agents automatically dissolve after losing an object for a timeout period. Since no inhibiting message will be heard, all sensors must remain in the idle state for this particular object, and a 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 subfields: 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.

(i) *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 this object.
(ii) *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 t.
(iii) *revoke_slave(s_i):* This is for a master to revoke its slave at sensor s_i.

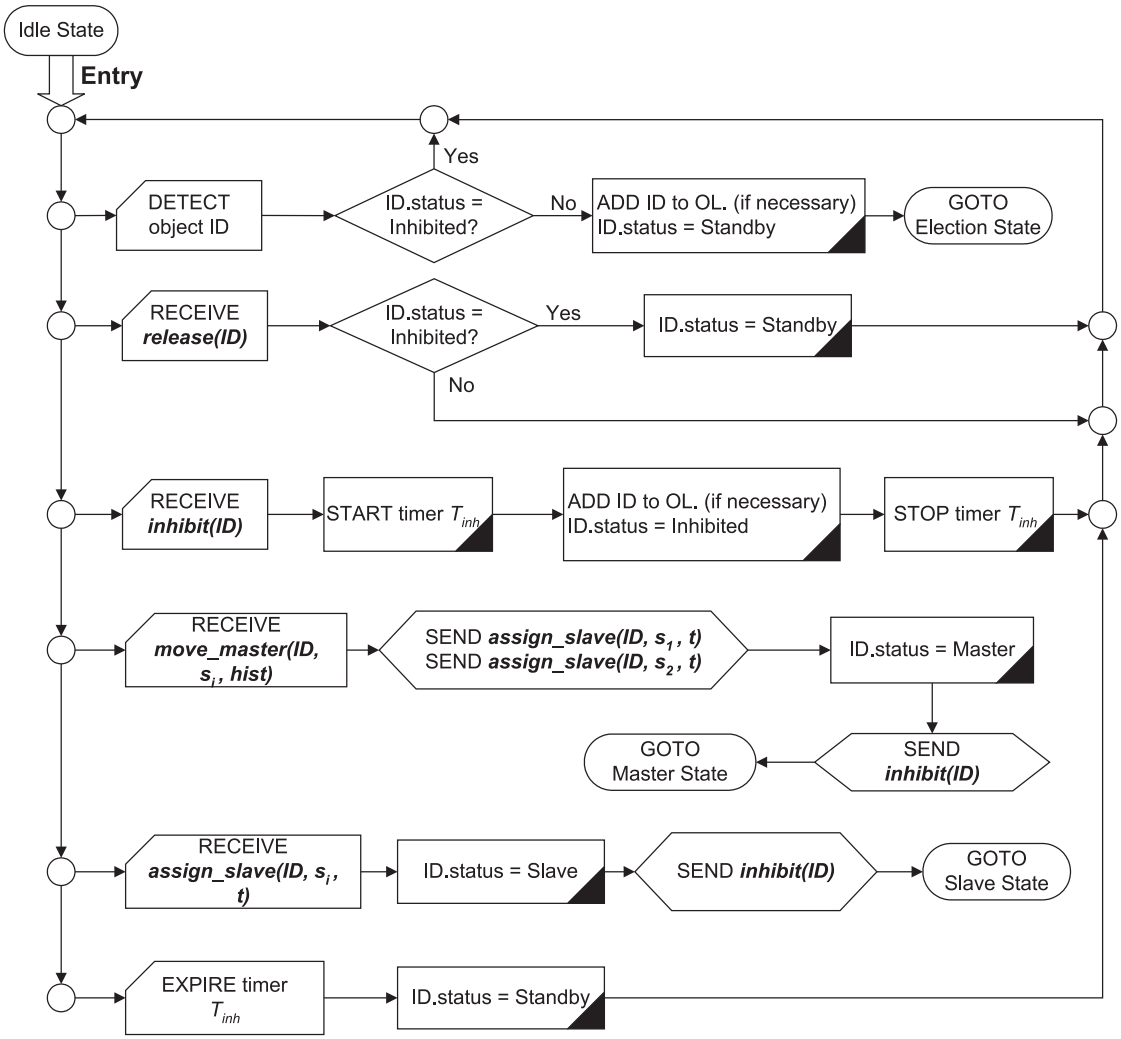

FIGURE 7. The Basic Protocol.

(iv) *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}.

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

(vi) *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.

(vii) *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.

Below, we formally present our four protocols. The Basic Protocol is shown in Figure 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 Figure 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 on different events, the sensor will go to the Master or Idle state.

Figure 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 Figure 2b. The last event is to inhibit irrelevant sensors from monitoring the object.

The Slave Protocol is shown in Figure 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.

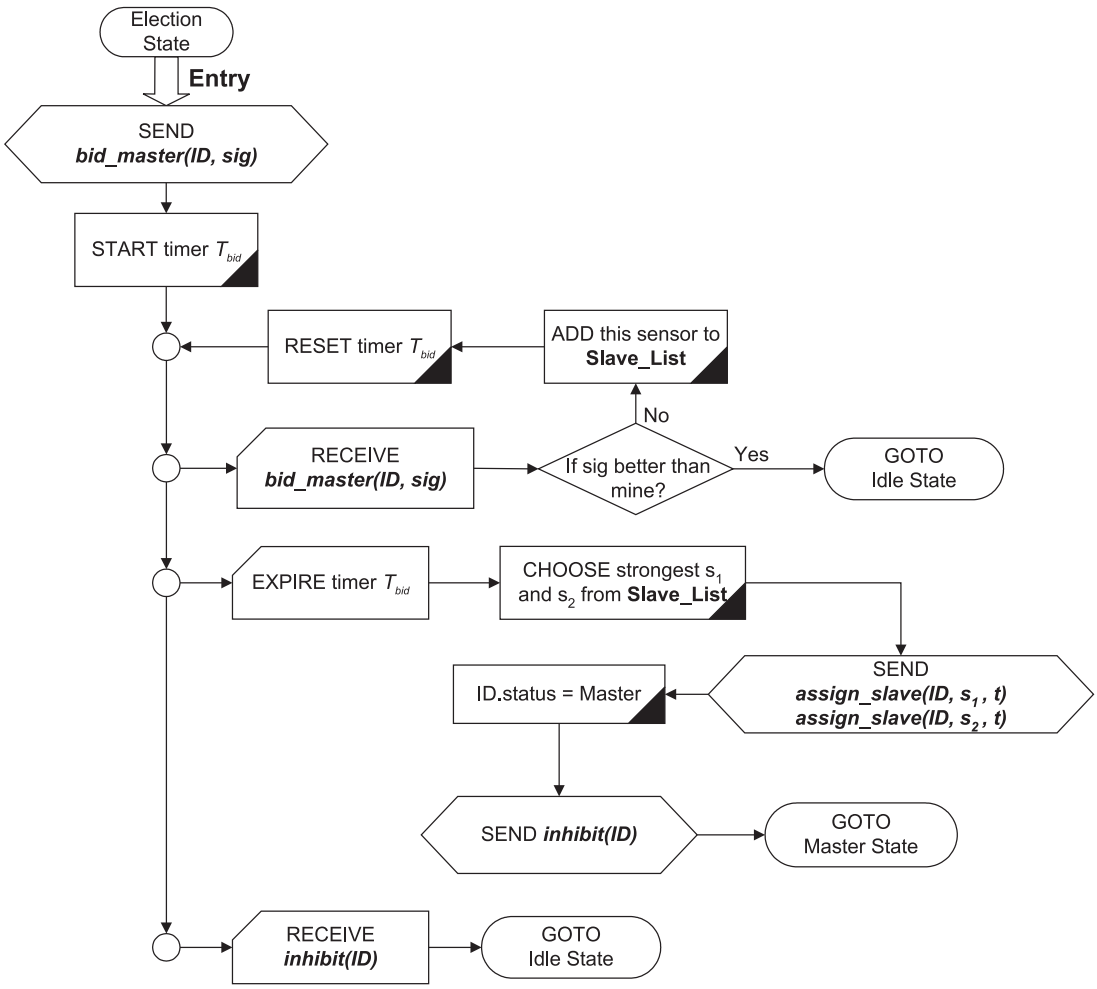

FIGURE 8. The Election Protocol.

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 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 a 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 [20]. 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 are 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. Figure 11a shows an example. The problem can be solved by a divide-and-conquer solution in time complexity $O(|V| \log |V|)$ [20].

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, Figure 11b 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 shown in Figure 11c.

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

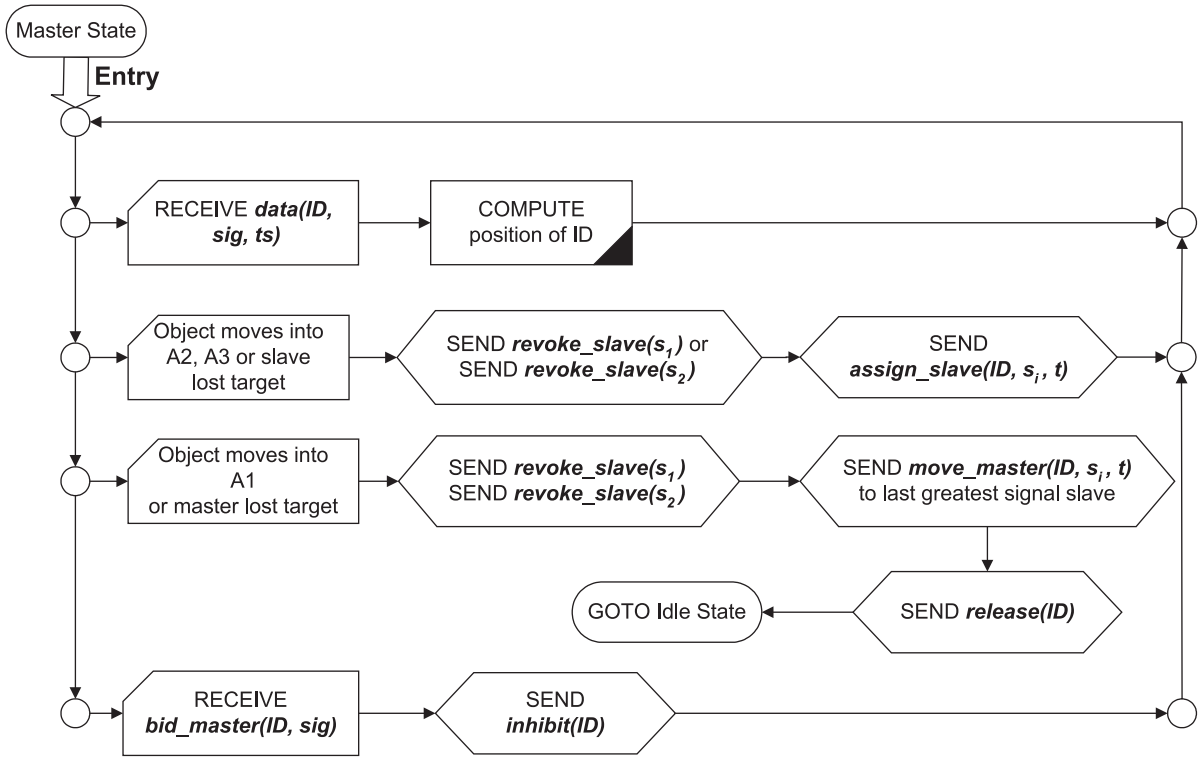

FIGURE 9. The Master Protocol.

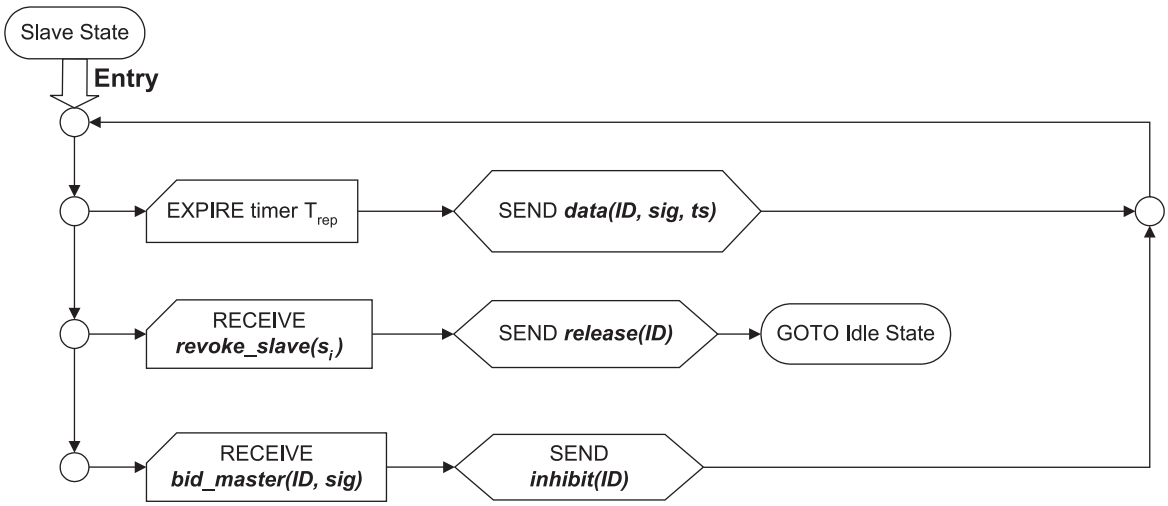

FIGURE 10. The Slave Protocol.

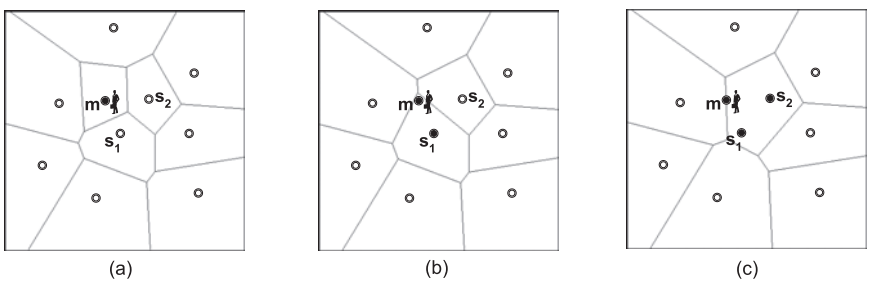

FIGURE 11. Using Voronoi graphs to find the master and slaves: (a) the Voronoi graph of all vertices, (b) the Voronoi graph after removing the master and (c) the Voronoi graph after removing the master and first slave.

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 computational time when 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 results 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 γ, and each tracking result is of size d bytes. That is, in a time interval Δt, the amount of tracking result is $\Delta t \cdot \gamma \cdot d$. Further, a sequence of tracking results can be combined with a fusion factor ρ, $0 \leq \rho < 1$, at a basic cost of b bytes. Specifically, the above tracking results can be compressed into $b + \Delta t \cdot \gamma \cdot d \cdot \rho$ bytes. In most cases, data fusion is beneficial. This normally happens when the data have a 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 for 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 when only the three sensors nearest to the object will track the object.

The second solution is called the 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 results 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 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 $\overline{T} \cdot \gamma \cdot d \cdot \rho$, 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. Note that the basic cost, b, is not needed since this cost is already involved in the previous tracking result (N_i). 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

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

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

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

- *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

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

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 Figure 12. The object can roam around and will measure beacon strengths transmitted from different

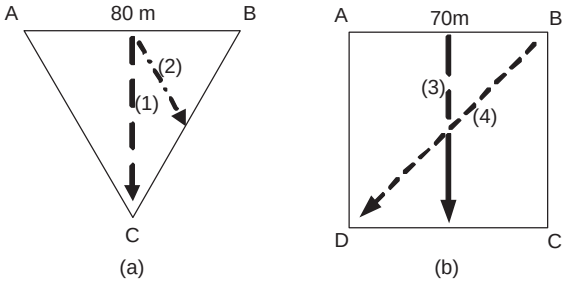

FIGURE 12. Experimental environment: (a) triangular sensor network and (b) square sensor network. Dashed lines represent tested roaming paths.

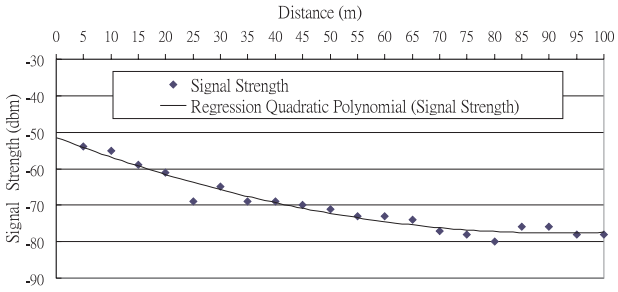

FIGURE 13. Experiment of signal strength versus distance for IEEE 802.11b.

sensors. For better accuracy, we average 10 samples in 1 s. It is worth noting that using the signal strength is only one of the possible methods of realizing the proposed idea. Other methods, such as ToA and TDoA, may also be applied.

First, we measure the degradation of signal strength versus distance. Figure 13 shows one set of data that we had collected. For every 5 m from 0 to 100 m 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 Figure 13. The curve is used to convert a received signal strength into an estimated distance.

Since signal strength is not an accurate measurement, the aforementioned trilateration algorithm cannot 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 Figure 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

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

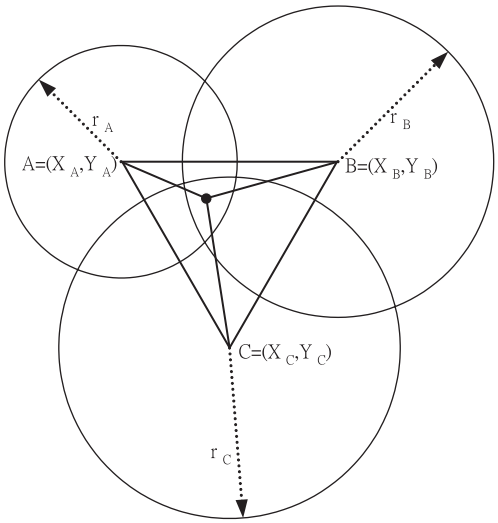

FIGURE 14. The position approximation algorithm.

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 m/s. 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 Figure 12a were tested. For roaming path (1), three sets of results are shown in Figure 15. For roaming path (2), the results are demonstrated in Figure 16. As can be seen, the predicted paths are close to the actual roaming paths, but there are still large gaps remaining that need to be improved further.

We have also tested the arrangement in Figure 12b, 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 Figures 17 and 18.

A larger-scale experiment with 12 sensors is shown in Figure 19a. 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). In contrast, if all sensors which can detect the object are allowed to track the object, the tracking result will be as shown in Figure 19b. 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 results in the same level of positioning accuracy.

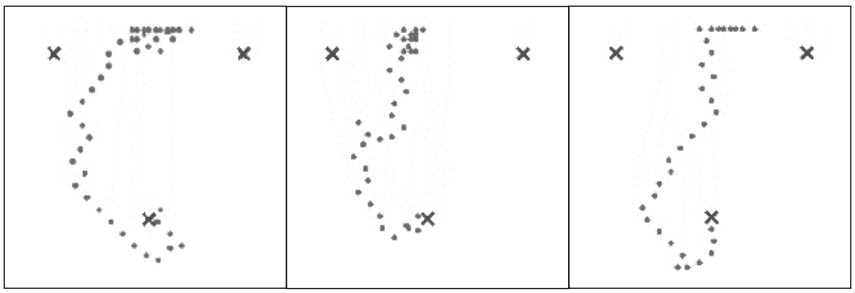

FIGURE 15. Tracking result of path (1) in Figure 12a.

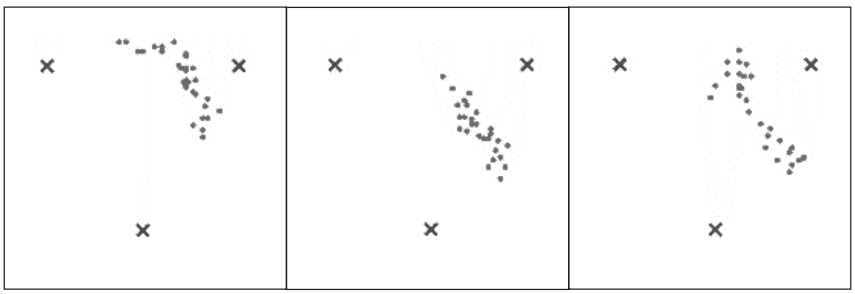

FIGURE 16. Tracking result of path (2) in Figure 12a.

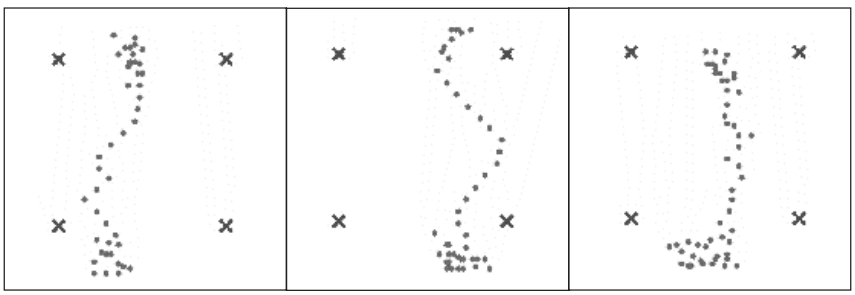

FIGURE 17. Tracking result of path (3) in Figure 12b.

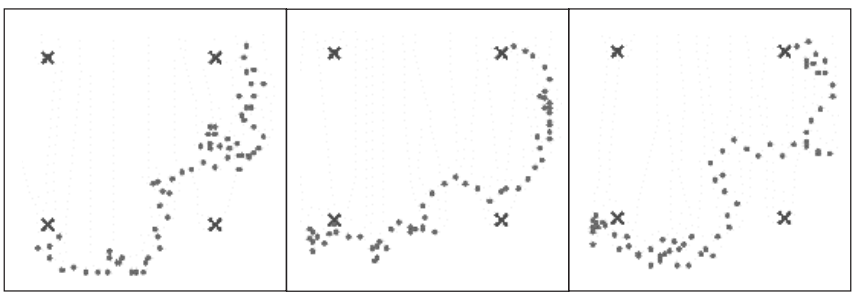

FIGURE 18. Tracking result of path (4) in Figure 12b.

5.2. Simulation results

To verify the advantage of using our agent-based approach, we have developed a simulator. Sensors are deployed in a $10{,}000 \times 10{,}000$ m^2 environment with triangular topology. The distance between two neighboring sensors is 80 m. The gateway is located at the center of the network. Each control packet is 2 bytes. Each location is represented by 2 bytes. Each routing header is equal to 2 bytes and each maximum transmission unit (MTU) can be as large as 500 bytes.

The Random Way Point Model [21] 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

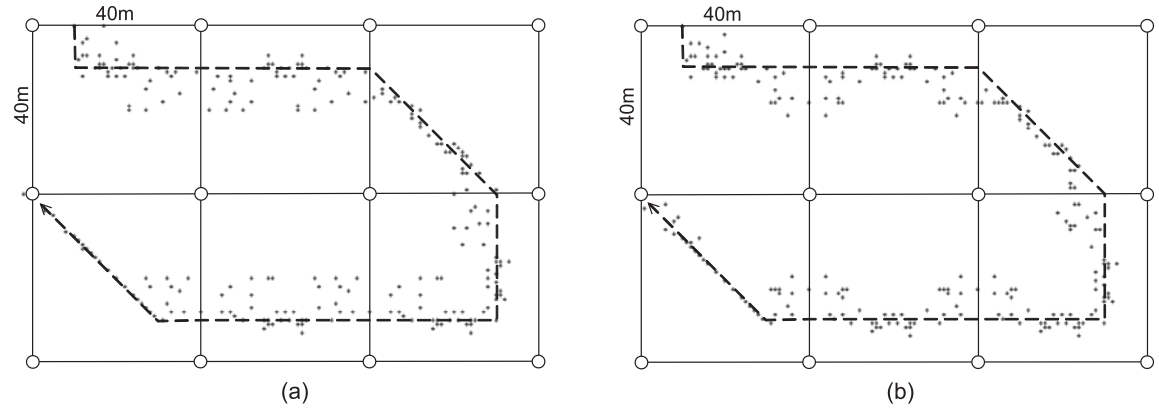

FIGURE 19. Comparison of tracking accuracy: (a) agent-based approach by using at most four sensors and (b) non-agent-based approach. Dashed lines are the real roaming paths and dots are the tracking results.

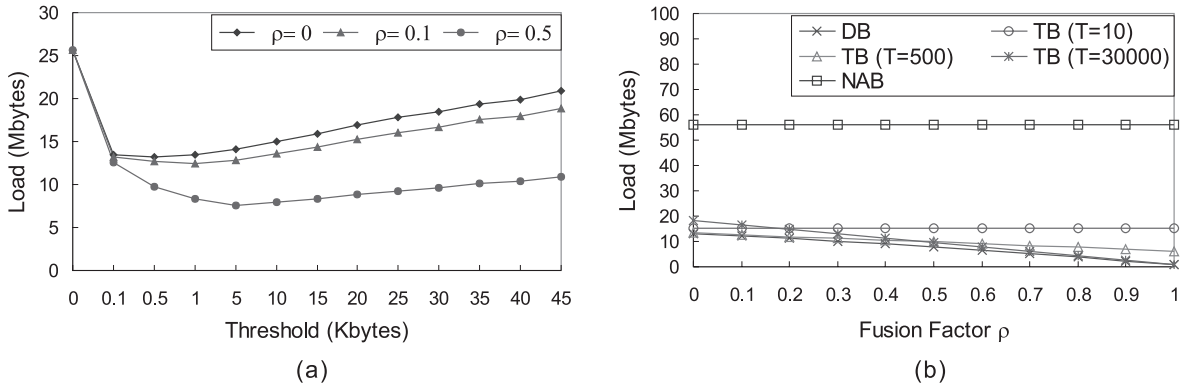

FIGURE 20. Simulation results: (a) the threshold T of TB versus traffic load and (b) the data fusion factor ρ versus traffic load.

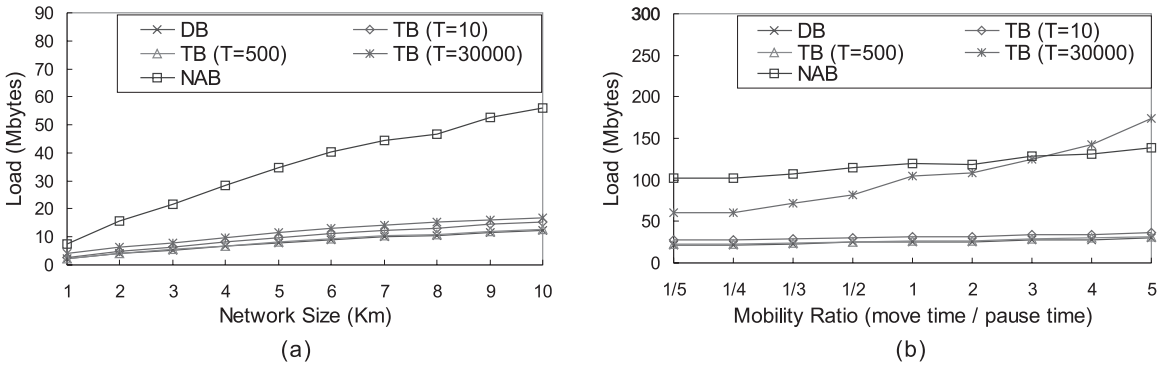

FIGURE 21. Simulation results: (a) network size versus traffic load ($\rho = 0.1$) and (b) mobility ratio versus traffic load ($\rho = 0.1$).

(x, y) is its current location. Note that locations outside the boundary are not considered. Under the moving state, the object moves at a constant speed of a uniform distribution between 1 and 3 m/s. After arriving at its destination, the object will pause for a period with an exponential distribution of mean $= 5$ s.

We first experiment on different threshold values of T for the TB strategy. The result is in Figure 20a. We measure the traffic load, which is defined to be the total data transmitted by all sensors (including relay traffic). A T significantly less than the largest MTU is not good due to high packet header overheads. In contrast, tuning T too large is also inefficient because the master agent will need to carry too much history while traveling. The figure suggests that a T equal to or slightly larger than the largest MTU would be a good choice. Figure 20b further demonstrates the effect of the fusion factor ρ. We compare different strategies. The DB strategy performs the best. The TB also performs very well, if a proper T can be selected. In all cases, NAB performs the worst.

In Figure 21a, we change the network size to visualize the effect. It is reasonable that larger networks incur higher traffic due to longer delivery paths. This justifies the importance of

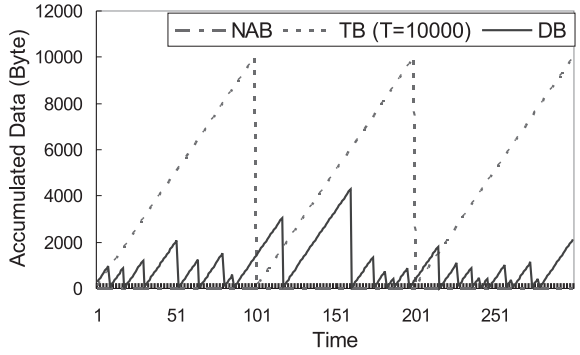

FIGURE 22. Size of accumulated data over time for different strategies.

using our agent-based strategies. In Figure 21b, 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 too large a threshold, TB will degrade significantly because the overhead for agents to carry tracking results would be significant as the mobility ratio increases.

Using agents to accumulate tracking results may cause some delay for the outside world to obtain the results. However, delay is difficult to calculate because it is related to many network characteristics (such as MAC, routing and transport protocols). So, we observe the amount of data in agents instead. Figure 22 shows the sizes of the accumulated tracking results over time for different strategies. NAB never accumulate data. TB has a regular (linear increasing) shape of accumulated size (here we used 10,000 as the threshold). DB's accumulated size is around 1235 in average.

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 60–80% 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 it has been proved that the prototyping works correctly, there is still room for improvement in the accuracy.

ACKNOWLEDGMENTS

Y. C. Tseng's research is co-sponsored by the MOE Program for Promoting Academic Excellence of Universities, Taiwan, under grant numbers A-91-H-FA07-1-4 and 89-E-FA04-1-4, by NSC of Taiwan under grant numbers NSC92-2213-E009-076 and NSC92-2219-E009-013, and by the Lee and MTI Center of NCTU.

REFERENCES

[1] Pottie, G. J. and Kaiser, W. J. (2000) Wireless integrated network sensors. *Commun. ACM*, **43**(5), 51–58.

[2] Savvides, A., Han, C. C. and Srivastava, M. B. (2001) Dynamic fine-grained localization in *ad-hoc* networks of sensors. In *Proc. Seventh Ann. ACM/IEEE Int. Conf. on Mobile Computing and Networking (Mobicom 2001)*, Rome, Italy, July, pp. 166–179. ACM Press.

[3] Savarese, C., Rabaey, J. and Beutel, J. (2001) Locationing in distributed *ad-hoc* wireless sensor networks. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2001)*, Salt Lake City, Utah, May, pp. 2037–2040. IEEE, Piscat, NJ.

[4] Estrin, D., Govindan, R., Heidemann, J. and Kumar, S. (1999) Next century challenges: scalable coordination in sensor networks. In *Proc. Fifth Ann. Int. Conf. on Mobile Computing and Networks (Mobicom 1999)*, Seattle, Washington, USA, August 15–19, pp. 263–270. ACM Press.

[5] Meguerdichian, S., Koushanfar, F., Qu, G. and Potkonjak, M. (2001) Exposure in wireless *ad-hoc* sensor networks. In *Seventh Ann. Int. Conf. on Mobile Computing ACM SIGMOBILE (Mobicom' 2001)*, Rome, Italy, July, pp. 139–150. ACM Press.

[6] Meguerdichian, S., Koushanfar, F., Potkonjak, M. and Srivastava, M. (2001) Coverage problems in wireless *ad-hoc* sensor networks. In *Proc. IEEE 2001, Conf. on Computer Communications*, Anchorage, AK, 22–26 April, pp. 1380–1387.

[7] Meguerdichian, S., Slijepcevic, S., Karayan, V. and Potkonjak, M. (2001) Localized algorithms in wireless *ad-hoc* networks: location discovery and sensor exposure. In *Proc. 2nd ACM Int. Symp. on Mobile Ad Hoc Networking and Computing, (MobiHoc 2001)*, Long Beach, CA, USA, October 4–5, pp. 106–116. ACM Press.

[8] O'Rourke, J. (1992) Computational geometry column 15. *Int. J. Comput. Geometry Appl.*, **2**(2), 215–217.

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

[10] Chakrabarty, K., Iyengar, S. S., Qi, H. and Cho, E. (2001) Coding theory framework for target location in distributed sensor networks. In *Int. Conf. on Information Technology: Coding and Computing (ITCC'01)*, Las Vegas, NV, USA, April 2–4, pp. 130–134.

[11] Horling, B., Vincent, R., Mailler, R., Shen, J., Becker, R., Rawlins, K. and Lesser, V. (2001) Distributed sensor network for real time tracking. In *Proc. 5th Int. Conf. on Autonomous Agents*, Montreal, June, pp. 417–424. ACM Press.

[12] Intanagonwiwat, C., Govindan, R. and Estrin, D. (2000) Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proc. Sixth Ann. Int. Conf. on Mobile Computing and Networks (Mobicom 2000)*, Boston, MA, August, pp. 56–67. ACM Press.

[13] Qi, H., Iyengar, S. S. and Chakrabarty, K. (2001) Multiresolution data integration using mobile agents in distributed sensor networks. *IEEE Trans. Syst. Man Cybernet. C*, **31**(3), 383–391.

[14] Enge, P. and Misra, P. (1999) Special issue on GPS: the global positioning system. *Proc. IEEE*, **87**, 3–15.

[15] van Diggelen, F. (2002) Indoor GPS theory & implementation. In *IEEE Position Location and Navigation Symp. (PLANS 2002)*, pp. 240–247.

[16] Rabaey, J., Ammer, J., da Silva, Jr, J. L. and Patel, D. (2000) PicoRadio: *ad-hoc* wireless networking of ubiquitous low-energy sensor/monitor nodes. In *Proc. IEEE Computer Society Ann. Workshop on VLSI (WVLSI'00)* Orlando, Florida, April, pp. 9–12.

[17] Bahl, P. and Padmanabhan, V. N. (2000) RADAR: an in-building RF-based user location and tracking system. In *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, March, pp. 775–784.

[18] Bulusu, N., Heidemann, J. and Estrin, D. (2000) GPS-less low-cost outdoor localization for very small devices. *IEEE Pers. Commun.*, **7**(5), 28–34.

[19] Fang, Q., Zhao, F. and Guibas, L. (2003) Lightweight sensing and communication protocols for target enumeration and aggregation. In *Fourth ACM Int. Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc'03)*, Annapolis, MD, USA, June 1–3, pp. 165–176. ACM Press.

[20] Aurenhammer, F. (1991) Voronoi diagrams: a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, **23**(3), 345–405.

[21] Johnson, D. B. and Maltz, A. D. (1996) Dynamic source routing in *ad hoc* wireless networks. In Imielinski, T. and Korth, H. (eds), *Mobile Computing*, Chapter 5, pp. 153–181. Kluwer Academic Publishers.