Distributed Energy-Efficient Target Tracking with Binary Sensor Networks

ZIJIAN WANG, EYUPHAN BULUT, and BOLESLAW K. SZYMANSKI
Rensselaer Polytechnic Institute

Target tracking is a typical and important cooperative sensing application of wireless sensor networks. We study it in its most basic form, assuming a binary sensing model in which each sensor returns only 1-bit information regarding target's presence or absence within its sensing range. A novel, real-time and distributed target tracking algorithm is introduced. The algorithm is energy efficient and fault tolerant. It estimates the target location, velocity, and trajectory in a distributed and asynchronous manner. The accuracy of the algorithm is analytically derived under an ideal binary sensing model and extensive simulations of ideal, imperfect, and faulty sensing models show that the algorithm achieves good performance. It outperforms other published algorithms by yielding highly accurate estimates of the target's location, velocity, and trajectory.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks*; C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Target tracking, binary sensor networks, energy efficient, distributed algorithms

ACM Reference Format:

Wang, Z, Bulut, E., and Szymanski, B. K. 2010. Distributed energy-efficient target tracking with binary sensor networks. ACM Trans. Sensor Netw. 6, 4, Article 32 (July 2010), 32 pages.
DOI = 10.1145/1777406.1777411 http://doi.acm.org/10.1145/1777406.1777411

32

This research was sponsored by U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence, or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Authors' addresses: Z. Wang (corresponding author), E. Bulut, B. K. Szymanski, Computer Science Department, Rensselaer Polytechnic Institute, Amos Eaton 109, 110 8th Street, Troy, NY 12180; email: wangz@cs.rpi.edu.

DOI 10.1145/1777406.1777411 http://doi.acm.org/10.1145/1777406.1777411

1. INTRODUCTION

Wireless sensor networks composed of miniature devices that integrate physical sensing, data processing, and communication capabilities present great opportunities for a wide range of applications [Chong and Kumar 2003]. Among them, target tracking is a representative and important application that usually requires a cooperative processing to achieve good results [Zhao et al. 2002; Brooks et al. 2003; Li et al. 2002; Rahman et al. 2007; Lin et al. 2006]. One of the fundamental studies of target tracking focuses on networks composed of sensor nodes capable of the most elementary binary sensing that provides just one bit of information about the target: whether it is present within the sensing range or not. These so-called binary sensor networks constitute the simplest type of sensor networks capable of target tracking [Arora et al. 2004; Aslam et al. 2003].

There are two kinds of binary sensing models for binary sensor networks: the ideal binary sensing model and imperfect binary sensing model. In the ideal binary sensing model, each node can detect exactly when the target falls into its sensing range R (as shown in Figure 1(a)). In the real world, detection ranges often vary depending on the environmental conditions, such as the relative orientation of the target and the sensor. These factors make target detection near the boundary of the sensing range much less predictable. The preceding observations give rise to an imperfect binary sensing model in which the target is always detected within an inner disk of radius R_{in} but it is detected only with certain nonzero probability in an annulus between the inner disk and an outer disk of radius R_{out}. Targets outside the outer disk are never detected (as shown in Figure 1(b)).

Although there are many papers about target tracking for wireless sensor networks, only a handful of research results on target tracking using binary sensor networks have been reported in recent years. The algorithms presented in Djuric et al. [2004] and Jing et al. [2007] first route the binary information to a central node and then the central node applies particle filters on information gathered from all sensors to update the target's track. Yet, particle filters are expensive to compute and transmitting data from each node to a central one is very costly in terms of the energy needed for communication for any nontrivial size network. In Mechitov et al. [2003], each point on the target's path is estimated by the weighted average of the detecting sensors' locations. Then, a line that best fits this point and the points on the trajectory established in the recent past are used as the target trajectory. Kim et al. [2005] improve the weight calculation for each sensor node that detected the target and use the estimated velocity to get the estimated target location. However, the last two methods require time synchronization of the entire network and assume that the target moves at a constant velocity on a linear trajectory. Furthermore, they only use positions of the sensor nodes that detected the target. Actually, the absence of detection can also provide information useful for improving tracking accuracy. In Shrivastava et al. [2006], both the presence and absence of the target within the node's sensing range are used to define local regions that the target had to pass. These regions are bounded by the intersecting arcs of the circles defined

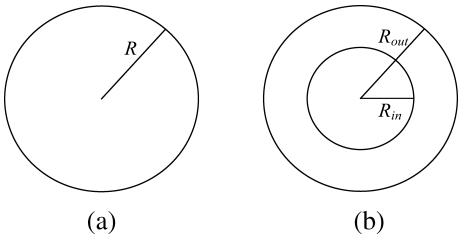


Fig. 1. Binary sensing models: (a) ideal, and (b) imperfect.

by the sensing ranges of the relevant nodes. The trajectory is estimated as a piecewise linear path with the fewest linear segments that traverses all the regions in the order in which the target passed them. However, the algorithm is centralized and complex to compute. It also requires a designated node to fuse data. Additionally, this designated node has to accumulate information from tracking sensors to form all regions needed to compute the estimated trajectory, which means that the tracking is not real time but delayed.

In this article, we introduce a novel distributed, energy-efficient, and fault-tolerant target tracking algorithm using binary sensor networks that applies to both ideal and imperfect binary sensing models. Each active node computes the target's location, velocity, and trajectory locally, but uses cooperation to collect the sensing bits of its neighbors. Furthermore, the algorithm tracks the target in real time, does not require time synchronization between sensor nodes, and can be applied to targets moving in random directions and with varied velocities. Moreover, the algorithm is tolerant to sensing faults, when a sensor either fails to detect a target within its range or reports a "phantom" target, as well as to information loss caused by packet collisions.

The remainder of the article is organized as follows. We describe the network model and our assumptions in Section 2. In Section 3, we introduce our target tracking algorithm in detail and describe its properties. In Section 4, we analytically derive the fundamental performance limits of our algorithm under the ideal binary sensing model. Section 5 presents the simulation results for both ideal and imperfect sensing models as well as for scenarios with faulty sensing and communications distorted by packet collisions. The conclusions are drawn in Section 6.

2. NETWORK MODEL AND ASSUMPTIONS

The sensor network comprises N nodes randomly uniformly distributed over a finite, two-dimensional planar region to be monitored. Each node has a unique identifier and the union of sensing regions of all network nodes guarantees redundant coverage of the monitored region. For simplicity, we assume that each node in the network has the same sensing range R under ideal binary sensing or the same radii R_{in} and R_{out} under the imperfect binary sensing model. However, our algorithm also applies when these ranges vary from node to node. Each node generates one bit of information ("1" for target's presence and "0" for its absence) only at the moment at which there is a change in the presence or

absence status of the target. Otherwise, we get no other information about the location, velocity, or other attributes of the target. To save energy and bandwidth, a node that has not detected change in the absence or presence of the target within its sensing range remains silent. Each time a new bit of information is generated, the node communicates it to its neighbors that are defined as nodes whose sensing ranges intersect its sensing range (depending on the relation of the sensing range to communication range, exchanging information with so-defined neighbors may require one-hop or multihop communication). We assume static (immobile) nodes. There are many practical examples of Unattended Ground Sensors (UGS) used in the military and security applications to make this case worthy of study on its own.

We also assume that each node knows its own location. This assumption can be satisfied by using some low-power GPS device or localization techniques (e.g., Savvides et al. [2001]). It should be noted that each sensor estimates the position of the target relative to its own location and sensing range. Hence, to report these results, sensors do not need to be aware of their geographical positions. Moreover, target velocity estimation requires only the knowledge of the positions of the neighbors relative to each other, which can be established via triangulation. Hence, establishing the geographical location of each node, although helpful, is not necessary for the proposed algorithm to work correctly. It is needed, though, for creating a central trajectory of the sensed target. Establishing location of neighbors by triangulation and then finding each node's geographical position based on some of those neighbors having GPS is a fairly standard procedure [Gentile 2007] that can be done at the network deployment, so it is not discussed here.

Nodes exchange their location information through communication at the network deployment stage. Each node has its own local timer and can time stamp sent or received messages. Additionally, we assume that the target moves with velocity that is low relative to the node's sensing frequency. Consequently, time of discovery of the change in the target's presence within the node's sensing range differs little from the time at which the target moves within or out of this range under the ideal binary sensing model.

3. TRACKING ALGORITHM

3.1 Basic Idea

To illustrate our basic idea, we use an example in Figure 2, which shows a target moving through an area covered by three nodes. Initially, the target is outside of the sensing ranges of all three nodes. Later, it moves within the sensing range of node N_x at time t_1, and then sensing ranges of nodes N_y at time t_2 and N_z at time t_3. Finally, it leaves sensing ranges of nodes N_x, N_y, and N_z, in that sequence, at times t_4, t_5, t_6, respectively.

According to the model described in the previous section, each node will generate a bit at the time of a change of the status of the target versus the sensor. It will generate bit "1" at the time of initial detection of the target's presence, and later bit "0" at the time of initial detection of its absence. Hence,

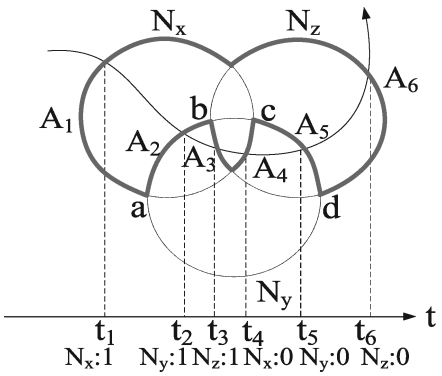


Fig. 2. An illustration of the basic idea behind the algorithm.

the messages are generated at the times at which the target enters and then exits the sensing range of the node. Thanks to the timing of this message, corresponding events are timed at each neighbor using its local clock, so the time differences between the events are correct even if the local clocks are skewed in terms of their measurements of the absolute time.

Clearly, at the transition time t_j, the target must be on arc A_j which is a part of the border circle of the sensing range of the node reporting the bit information. This arc can be determined cooperatively from the presence and absence bits of neighbors of that node. Let us consider arc A_2 defined at time t_2 as an example. At time t_2, node N_y senses the target presence within its sensing range for the first time, so the arc is a part of the sensing range border circle of node N_y. At that time, node N_y knows that the target is within the sensing range of node N_x, so the target must be on arc "abc". Node N_y knows also that the target is not within the sensing range of node N_z, so the target cannot be on arc "bcd". Hence, node N_y concludes that the target must be on arc A_2. An important observation is that, by using this method, the two-dimensional uncertainty of the target's location on the plane is reduced to a one-dimensional uncertainty within the circle section. The shorter this circle section is, the smaller the uncertainty becomes.

3.2 Tracking Algorithm under Ideal Binary Sensing Model

The initial idea of the tracking algorithm under the ideal binary sensing model was presented in Wang et al. [2008a] and it can be summarized as follows. At the network deployment stage, each node initializes the list of statuses of its neighbors to "0"s. Each time a node receives one-bit information from a neighbor, it updates its status on the list. At the moment at which the node discovers the change in the target's presence within its sensing range, it identifies the arc of its sensing range border circle that the target is crossing. The target location is estimated as the middle point of the corresponding arc and broadcasted to neighbors. Two relatively accurate estimates of target location combined with the difference of local times at which these estimates were made are used for distributed computation of the target velocity. A weighted line fitting method

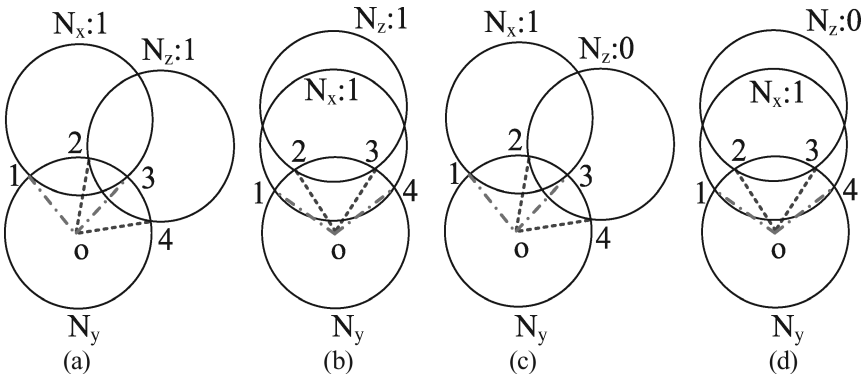


Fig. 3. Instances of angle combinations.

is used to find a line, approximating a fragment of the target trajectory, that best fits the estimated target locations.

3.2.1 *Initialization and Information Update.* In the initialization procedure, each node establishes a list of its neighbors. Each element of the list stores the following information: neighbor node identifier, intersection points of the sensing circles of the node and its neighbor, an angle corresponding to the arc defined by these intersection points, and one-bit information generated by the neighbor, initialized to "0". Upon receiving one-bit information from a neighbor, the node updates the corresponding entry in the list.

3.2.2 *Location Estimate.* We combine all angles corresponding to arcs defined by the neighbor list to determine the arc that the target is crossing. The four instances of this process are shown in Figure 3. If the neighbors both generated bits equal to "1", the corresponding central angles are combined by "&" operation that returns the intersection of these two angles. As shown in Figure 3(a), the common angle of $\angle 1o3$ and $\angle 2o4$ is $\angle 2o3$, so the node N_y estimates the target location as the middle point of arc "23" when it senses that the target just moved within its sensing range. One special instance is shown in Figure 3(b), where the common angle is just one of the two angles.

If one neighbor status is set to "1" while the other is set to "0", the corresponding central angles are combined with the "-" operation that returns the angle formed from the first angle by excluding the second angle from it. For example, in Figure 3(c) $\angle 1o3 - \angle 2o4$ is equal to $\angle 1o2$. In a special case shown in Figure 3(d), the result may consist of two angles, $\angle 1o2$ and $\angle 3o4$. The correct angle in this case is chosen by considering the recent estimate of the target location.

Let *FA* be the sought arc's central angle initialized to 2π (the entire circle of the sensing border of a node). Let *IN* be the set of neighbor nodes with status set to "1" and let *OUT* be the set of neighbor nodes with status set to "0". Then, the final angle whose corresponding arc is the one that the target is crossing

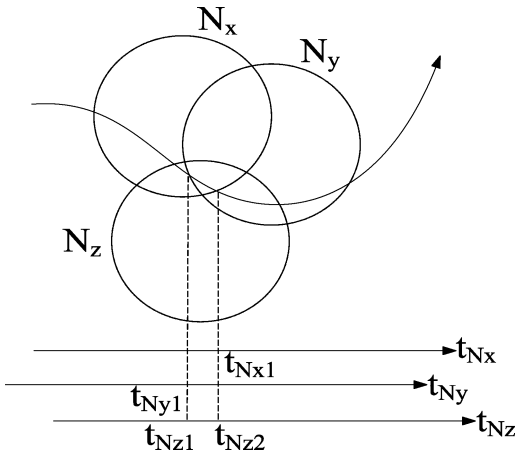


Fig. 4. Velocity estimation.

can be expressed as

$$FA = FA \underset{i \in IN}{\&} angle_i \underset{j \in OUT}{-} angle_j, \tag{1}$$

where $angle_i$ is the central angle corresponding to neighbor i.

3.2.3 *Velocity Estimate.* We use a distributed, asynchronous algorithm to estimate the target velocity. As shown in Figure 4, three nodes N_x, N_y, and N_z work asynchronously. At time t_{Ny1} on node N_y's local clock, node N_y senses the target's presence within its sensing range for the first time and generates a bit "1" message. The estimated location of the target is also included in this message to save energy and bandwidth. Since the elapsed time of radio transmission is negligible, node N_z receives this message at time t_{Nz1} on its local clock. Node N_z will also receive the message from node N_x at time t_{Nz2}. Then, node N_z can use the time difference $t_{Nz2} - t_{Nz1}$ and the difference of locations reported in these two messages to estimate the target velocity. To estimate velocity accurately, only location estimates with relatively high accuracy are used; these are locations at the middle points of the short arcs.

3.2.4 *Trajectory Estimate.* A weighted line fitting method is used to get the target trajectory and the weight of each estimate is defined as

$$w = \frac{|circle|}{|arc|}, \tag{2}$$

where $|arc|$ is the length of corresponding arc whose middle point is the estimated target location and $|circle|$ is the length of the sensing range border circle. Each node finds the line that best fits these weighted estimated locations. This line, when expressed as $y = a \cdot x + b$, minimizes the metric Q defined as

$$Q = \sum_{i \in E} w_i (y_i - a \cdot x_i - b)^2, \tag{3}$$

where $E = [(y_0, x_0), \ldots (y_i, x_i), \ldots (y_k, x_k)]$ is the list of the estimated target locations to which the line is fitted. Hence, this is the weighted least square error line.

Each node will cache k (an adjustable parameter) estimated locations received from neighbor nodes. The node will divide these estimated locations into groups of size n (another adjustable parameter, a divisor of $k + 1$) from the beginning of the cache and use weight line fit method on the first group of estimated locations to get a line segment with slope k_1. Then the node will get new line segment with slope k_1' using next group of estimated locations. If the difference between the slopes is less than ϵ (the third adjustable parameter), we assume that there is no turn in the trajectory and a new slope of line segment will be calculated using the next group of estimated locations. Otherwise, the node will select the current group as an new start to get a new line segment with slope k_2. This procedure will continue to the end of the cached locations. The specific values of the three parameters k, n, ϵ used in our simulations are given in Section 5.5.

3.3 Tracking Algorithm under Imperfect Binary Sensing Model

To make our algorithm robust, as in Shrivastava et al. [2006], we take a worst-case approach to the information provided by the imperfect binary sensing model: if a sensor output is "1", then we assume that the target is somewhere inside the large disk of radius R_{out}; if a sensor output is "0", then we assume that the target is somewhere outside the small disk of radius R_{in}.

The initial investigation of this case was given in Wang et al. [2008b] and it showed that the main influence of the imperfect binary sensing model is that the algorithm no longer could identify circular arcs that the target crosses (as was possible in the ideal binary sensing model) when there the status of the target sensed by a node changes. Instead, we can only identify that the target must be within the ring determined by R_{in} and R_{out}. However, we can use a thin ring section which is determined by the neighbor output to approximate the circular arcs and then estimate the position of the target. Although this will make the one-dimensional uncertainty of the target's location expand to a two-dimensional uncertainty, if the resulting ring section is short and thin, the error will still be small.

3.3.1 *Initialization.* In the initialize procedure, each node establishes a list of its neighbors and calculates the exact angle corresponding to a neighbor depending on the output and the relative position of that neighbor.

The three instances for neighbor (node N_y) that outputs bit "1" are shown in Figure 5. As described previously, if node N_y outputs bit "1", we can only be sure that the target is within sensing range R_{out}. When node N_x senses there is a change in the status of the target, it knows that the target is within the ring determined by R_{in} and R_{out}. Depending on the relative position of node N_y to node N_x, there would be up to two angles corresponding to node N_y resulting from the intersection of R_{in} and R_{out} circle of node N_x and R_{out} circle of node N_y. If two angles exist for node N_y, we choose the angle that ensures that the target

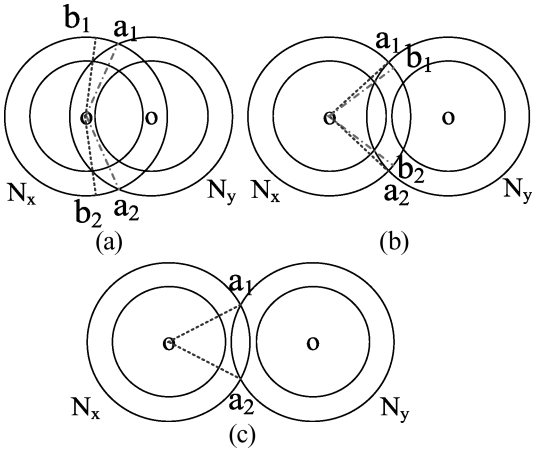


Fig. 5. Determination of an angle corresponding to neighbor's output "1".

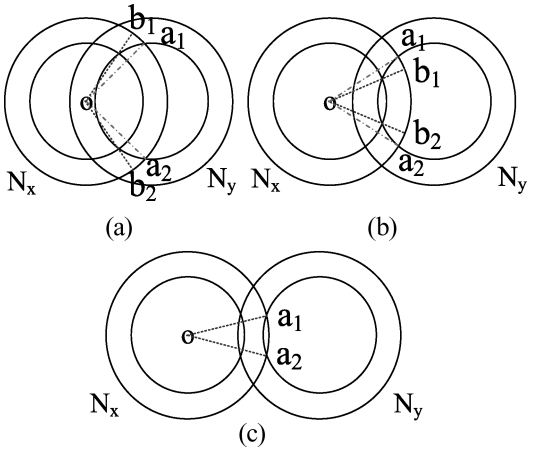


Fig. 6. Determination of an angle corresponding to neighbor's output "0".

must fall in this angle, for example, we choose $\angle b_1ob_2$ and $\angle a_1oa_2$ in Figure 5(a) and 5(b) as the angle corresponding to neighbor N_y. If only one angle exists for node N_y, then it is chosen as the corresponding angle, as shown in Figure 5(c).

The three instances for neighbor (node N_y) that outputs bit "0" are shown in Figure 6. As described previously, if node N_y outputs bit "0", we can only know that the target is outside sensing range R_{in}. Depending on the relative position of node N_y to node N_x, there would be up to two angles corresponding to node N_y resulting from the intersection of R_{in} and R_{out} circle of node N_x and R_{in} circle of node N_y. If there are two existing angles for node N_y, we choose the angle that ensures that the target must fall out of this angle, for example, we choose $\angle a_1oa_2$ and $\angle b_1ob_2$ in Figure 6(a) and 6(b) as the corresponding angle to neighbor N_y. For the instance shown in Figure 6(c), we cannot determine that the target is outside $\angle a_1oa_2$ because the target could be within $\angle a_1oa_2$ no matter

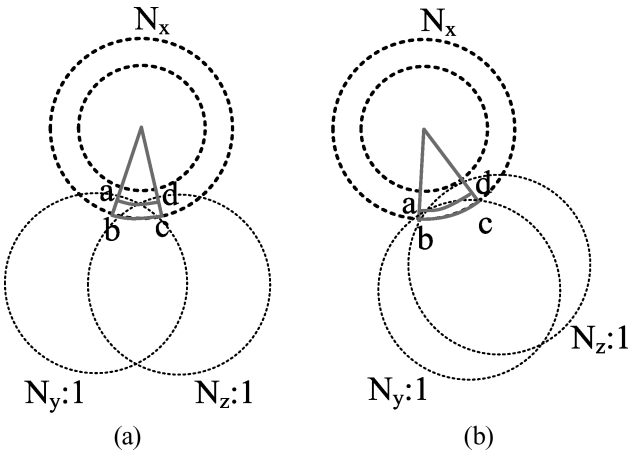


Fig. 7. Ring section thickness calculation.

what node N_x outputs. So, if node N_y outputs bit "0", it will be considered as a neighbor of node N_x only if its R_{in} circle intersects with the R_{in} circle of node N_x.

3.3.2 *Location Estimate.* At the moment at which the node discovers the change in the target's presence, it calculates the final angle corresponding to the ring section that the target is crossing using the same angle combination method as in the ideal binary sensing model. Then, the thickness of the ring section is recalculated to make the estimation of target position more accurate.

We calculate the intersection points of each pair of node N_x's neighbor that output bit "1". The intersection point that falls into the final angle and is furthest away from the center of node N_x determines one of the boundaries of the ring section, which will make the ring section as thin as possible. A new thinner ring section is determined by this intersection point, R_{out} circle, and final angle. For the case shown in Figure 7(a), this is ring section "abcd". Interestingly, the neighbor node that outputs bit "0" contributes only to the angle combination but not to the thickness calculation. As shown in Figure 7(b), the recalculated ring section may exclude some area into which the target may fall, although with small probability because this area is near R_{out} circle of node N_x. Moreover, when the final angle is small, this area will be negligible in size. The target position is estimated as the center point of this ring section.

3.3.3 *Velocity Estimate.* We use the same method as under the ideal binary sensing model to estimate the target velocity. At the time of sending out the message that contains the estimate position, the target may not be exactly at the boundary of sensing circle, as was the case under the ideal binary sensing model shown in Figure 4. Hence, to estimate velocity accurately, only location estimates with relatively high accuracy are used for which the ring sections are short and thin.

3.3.4 *Trajectory Estimate.* We use the same method as under the ideal binary sensing model to estimate the target trajectory but redefine the weight of each estimate as

$$w = \frac{|ring|}{|ring\ section|}, \tag{4}$$

where $|ring\ section|$ is the area of the corresponding ring section whose middle point is the estimated target location and $|ring|$ is the area of the ring determined by circles R_{in} and R_{out}.

3.4 Properties of Our Tracking Algorithm

The most prominent feature of our algorithm is its energy efficiency supported by its four inherent properties. The first such property is the use of binary sensing that often reduces sensing energy requirements. For many types of sensors, a binary detection uses either a simple threshold mode or on-board signal processing, both of which reduce power consumption significantly. As an example, for acoustic sensing (e.g., the KnowlesEA-21842 sensor) and magnetometer sensing (e.g., the Honeywell HMC1002 sensor), using binary mode reduces power consumption five-fold compared to what a classification mode requires [Singh et al. 2007].

The second property is that each node executing our algorithm generates a message only when there is a change in the target status. Only two relatively short messages are generated over the entire period during which the target resides within the sensing range of a node.

The third property is processing the target information in a distributed manner without routing the target information from each node back to the central node which would consume a considerable amount of energy. Finally, as we will show in Section 5.3.2, our tracking algorithm achieves higher accuracy with smaller average number of neighbors than other binary sensing target tracking algorithms. Having fewer active nodes saves a lot of energy and prolongs the network lifetime.

Another interesting property of our algorithm is that the reporting and not reporting the presence of the target in the sensing range by the neighboring nodes can be verified for consistency against the known topology of the neighbor graph. In case of perfect binary sensing, all neighbors reporting presence of the target must have nonempty intersection of their sensing ranges. Moreover, none of the sensors not reporting presence of the target can have such an intersection entirely within its sensing range. We used this property to provide a level of tolerance to reporting faults in our algorithm. The details of the modification and measurements of the resulting fault tolerance are described in Section 5.6. This is important because there are many reasons a node may fail to report the target presences within its sensing range, or conversely to report the target that is absent from its sensing range. Both kinds of errors will influence the target position estimation. A typical reason for such errors is environment noise that may affect the sensing device of a sensor [Liu et al. 2007]. Also any failure of communication (e.g., a collision of packets carrying the reports) will result in

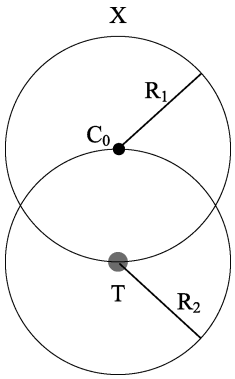


Fig. 8. A configuration for performance analysis.

the new target status not being updated. Hence, fault tolerance to reporting faults improves the algorithm performance in real applications.

4. FUNDAMENTAL PERFORMANCE LIMITS UNDER IDEAL BINARY SENSING MODEL

Assume that we have a domain of area A_d in which there are total of N sensors, each with a uniform sensing range (to simplify the analysis, we set this sensing range to be one unit, $R = 1$). Let us consider the specific time instance at which the target T has been just sensed by node X whose center is at point C_0. All neighbor sensors that also sense the target are within a circle centered at T with radius of $R_2 = 1$ unit, as shown in Figure 8. According to the algorithm introduced in Section 3, the accuracy of our algorithm under the ideal binary sensing model is just half of the length of the arc resulting from the angle combination procedure, which is the length of the average shortest arc from target T to the intersection point resulting from the intersection between the sensing circle of neighbor node and the sensing circle of the node X. First, we analyze the accuracy of our algorithm considering only the output from neighbor nodes that also sense the target. Then, we extend this analysis to the case in which the output from neighbor nodes that do not sense the target is also considered.

The probability P_k that there are $k\,(0 \leq k < N)$ neighbors that also sense the target within their sensing ranges is

$$P_k = \binom{N-1}{k} \left(\frac{\pi}{A_d}\right)^k \left(1 - \frac{\pi}{A_d}\right)^{N-k-1}. \tag{5}$$

A practical way to compute this probability is to start with the most probable number of neighbors, which is

$$k_{most} = \left\lceil (N-1)\frac{\pi}{A_d} \right\rceil. \tag{6}$$

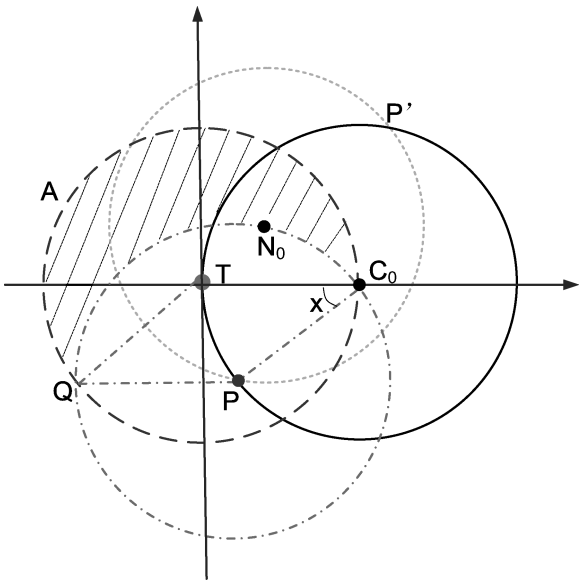


Fig. 9. A configuration for calculating the shared arc length distribution.

Then, starting with $k_{most} - 1, k_{most} + 1$, the probabilities for other k's can be calculated by the following formulas.

$$P_{k+1} = \frac{(N-k-1)\pi}{(k+1)(A_d - \pi)} P_k; \quad P_{k-1} = \frac{k(A_d - \pi)}{(N-k)\pi} P_k \tag{7}$$

Once the values of P_{k+1} become very small, the computation of the subsequent P_k probabilities stops.

All k intersecting arcs will have one point on one side of the target and another on the other side. So the resulting average shortest length of the arc of all intersections will be the double of the length of the distance of the target from the closest intersection point on the either side of the target. Then, the accuracy of our algorithm will be just half of the average shortest length of the arc. Based on this insight, we introduce the method of calculating the distribution of the length of the shared arc, as shown in Figure 9. Node X with sensing range of one unit centered at point C_0 (we refer to this circle as circle C_0) just senses the target T. Node Y centered at point N_0 is one of the neighbors of node X that also senses the target within the sensing range of one unit (we refer to this circle as circle N_0). These two sensing circles intersect at points P and P'. We chose the notation so that P is the closest of the two intersection points to the target and its distance may determine the accuracy of the target position measurement. All the neighbors of node X that also sense the target must fall in the circle centered at point T with radius of one unit (we mark this circle as circle T). It is important to note that any node whose sensing circle also intersects with circle C_0 at point P must be on the circle centered at point P with radius of one unit (we refer to this circle as circle P). Consequently, any node that also senses the target but has an intersection point with circle C_0

closer to target T than point P must fall in the shadowed area that we denote as A. If x denotes $\angle PC_0T$ size in radians, then the length of arc PT is $1 \times x = x$. The probability $P(x)$ that the length of arc PT is less than or equal to x is $\|area\ A\|/\|circle\ T\|$, where $\|\ldots\|$ returns the area of its argument, sector QTC_0 is in circle T, and sector QPC_0 is in circle P.

$$\begin{aligned}\|area\ A\| &= \|sector\ QTC_0\| - \|sector\ QPC_0\| + \|QTC_0P\| \\ &= \pi \times R^2 \times (2\pi - (\pi - x))/2\pi - \pi \times R^2 \times (\pi - x)/2\pi \\ &+ 2 \times 1/2 \times (2 \times R^2 \times \cos(x/2)) \times \sin(x/2) = (x + \sin(x)) \times R^2 \end{aligned} \tag{8}$$

Hence, $P(x) = x + \sin(x)/\pi$ and the probability $P_s(x)$ for the shortest arc created by k neighbor nodes sensing the target being shorter than x is defined as [Feddema et al. 1999]

$$P_s(x) = 1 - Prob(y_s \geq x) = 1 - \prod_{i=1}^{k} Prob(y_i \geq x) = 1 - (1 - P(x))^k. \tag{9}$$

Hence, the average length of the shortest arc is

$$\delta = \sum_k P_k \int_{x=0}^{\pi} x d(1 - (1 - P(x))^k) \approx \sum_{k=0}^{k_{\max}} P_k \delta_k, \tag{10}$$

where

$$\delta_k = \int_{x=0}^{\pi} -x d(1 - P(x))^k = \int_{x=0}^{\pi} \left(1 - \frac{x + \sin(x)}{\pi}\right)^k dx. \tag{11}$$

We select $k_{\max}$ in such a way that $P_{k_{\max}+1} < 0.1\% \leq P_{k_{\max}}$.

In the previous procedure, we only consider nodes that sense the target. Yet, nodes that do not sense the target also contribute to the accuracy of the algorithm. Let AB denote the average shortest arc with length of $2L$ centered at point T obtained through the preceding procedure and shown in Figure 10. All nodes that intersect with circle C_0 must fall within interior of the circle centered at point C_0 with radius of two units. Among all of these nodes, the ones that do not sense the target but also have one intersection points on arc AT must fall in the shadowed area which we will refer to as W and which is formed by the circles of unit radius centered at points A and T. The nodes within area W also contribute to the accuracy of estimation of the target position by cutting the feasible arc shorter.

Similar to the previous analysis, if one node falling in area W intersects with arc AT at point P, then any node that does not sense the target but has an intersection point Q closer to point A than P must fall in the shadowed area shown in the right-hand side of Figure 10 that we will denote as Z.

Let $k_{sensing}$ denote the number of sensors that sense the target at the given time instance. We assume that this number includes the node that just observed the target in its sensing range and its neighbors who also have the target in their sensing ranges. Then the probability P'_k that there are k ($0 \leq k \leq N - k_{sensing}$) neighbors that do not sense the target within their sensing ranges

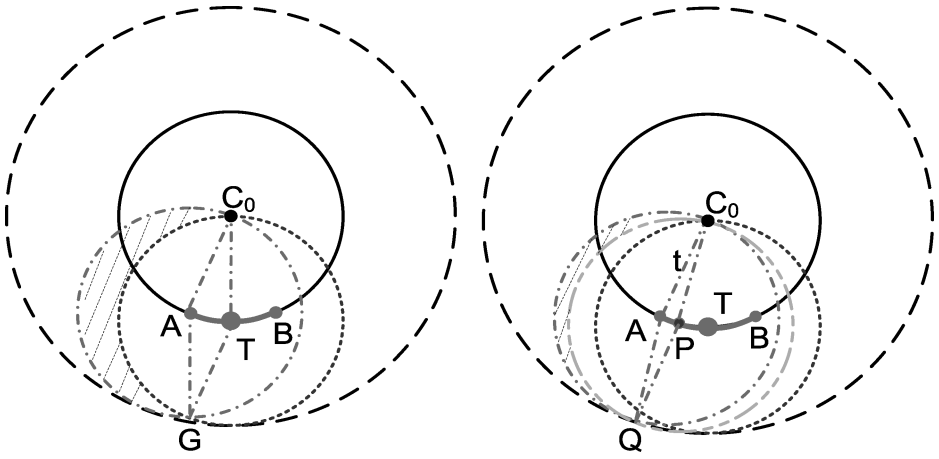


Fig. 10. Illustrations of contributions of the nodes that sense no target and distribution of the cut-off arc length.

but contribute to the accuracy is

$$\begin{aligned} P'_k &= \binom{N-k_{sensing}}{k} \left(\frac{\|area\ W\|}{A_d-\pi}\right)^k \left(1-\frac{\|area\ W\|}{A_d-\pi}\right)^{N-k-k_{sensing}} \\ &= \binom{N-k_{sensing}}{k} \left(\frac{L+\sin(L)}{A_d-\pi}\right)^k \left(1-\frac{L+\sin(L)}{A_d-\pi}\right)^{N-k-k_{sensing}}. \end{aligned} \tag{12}$$

A practical way to compute this probability is to start with the most probable number of neighbors, which is

$$k'_{most} = \left\lceil (N-k_{sensing})\frac{L+\sin(L)}{A_d-\pi} \right\rceil. \tag{13}$$

Then, the probability for other k's can be calculated by

$$P'_{k+1} = \frac{(N-k-k_{sensing})(L+\sin(L))}{(k+1)(A_d-L-\sin(L)-\pi)} P'_k \tag{14}$$

$$P'_{k-1} = \frac{k(A_d-L-\sin(L)-\pi)}{(N-k+1-k_{sensing})(L+\sin(L))} P'_k. \tag{15}$$

Once the values of P'_{k+1} become very small, the computation of the subsequent P'_k probabilities stops.

If we denote $\angle PC_0A$ as t in radian, then length of arc PA is $1 \times t = t$. The probability $P'(t)$ that the length of arc PA is less than or equal to t is

$$P'(t) = \|area\ Z\|/\|area\ W\| = (t+\sin(t))/(L+\sin(L)). \tag{16}$$

The probability $P'_l(t)$ that the longest arc created by k neighbors not sensing the target is shorter than t is defined as

$$P'_l(t) = Prob(y_l \le t) = \prod_{i=1}^{k} Prob(y_i \le t) = P'(t)^k. \tag{17}$$

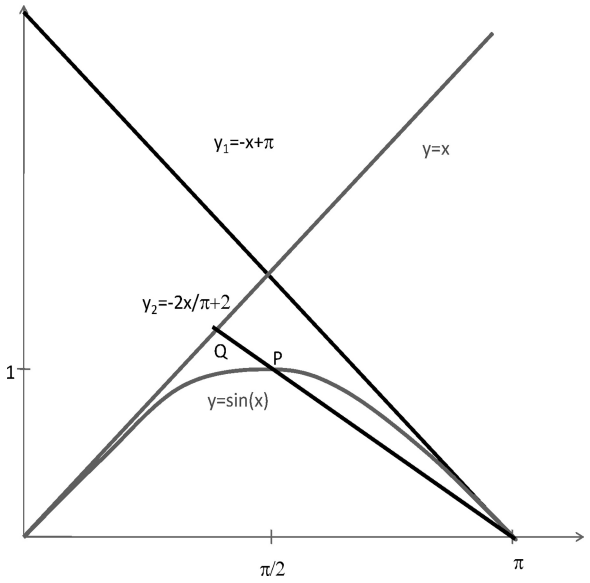


Fig. 11. Analysis of the algorithm's error for nearly uniform sensor distribution.

Hence, the average length of the longest arc is

$$\delta'(L) = \sum_j P'_j \int_{t=0}^{L} t dP'(t)^j = L - \sum_j P'_j \delta'_j, \tag{18}$$

where

$$\delta'_j(L) = \frac{\int_{t=0}^{L} (t + \sin(t))^j dt}{(L + \sin(L))^j}. \tag{19}$$

We select $k'_{\max}$ in such a way that $P'_{k'_{\max}+1} < 0.1\% \leq P'_{k'_{\max}}$. With this notation, the accuracy of our algorithm is

$$\epsilon = \sum_k P_k \delta'(\delta_k) \approx \sum_{k=0}^{k=k_{\max}} P_k \sum_{j=0}^{j=k'_{\max}} \frac{P'_j}{(\delta_k + \sin(\delta_k))^j} \int_{t=0}^{\delta_k} (t + \sin(t))^j dt. \tag{20}$$

Let us consider now the nearly uniform distribution of sensors with the density ρ_n measured as the average number of sensors in the unit square when the sensing range $R = 1$. Consider a node whose sensing boundary is crossed. With the preceding notation, on average there are $k = \pi R^2 \rho_n - 1 = \pi \rho_n - 1$ other sensor nodes that sense the target at that time. Since we assume nearly uniform distribution of sensors, then, with probability one, we will have k nodes sensing the target and the entire Eq. (10) becomes just $\delta_k = \int_{x=0}^{\pi} (1 - (x + \sin(x))/\pi)^k dx$. Setting $y = \pi - x$ transforms $\pi - x - \sin(x)$ into $y - \sin(y)$ and then $\delta_k =$

$\int_{y=0}^{\pi}(y-\sin(y))^k dy/\pi^k$. From Figure 11, we get

$$\int_{y=0}^{\pi}(y-\sin(y))^k dy > \int_{y=\pi/2}^{\pi}(2y-\pi)^k dy = \frac{\pi^{k+1}}{2(k+1)} \tag{21}$$

hence the error in radians is larger than $\frac{\pi}{2(k+1)}$.

On the other hand, for line y_2, coordinates of point Q are $(\frac{2\pi}{2+\pi}, \frac{2\pi}{2+\pi})$ so the area above $\sin(x)$ over the interval $[0, \pi/2]$ is

$$2\left(\frac{\pi}{2+\pi}\right)^2 + \left(\frac{\pi}{2+\pi}+\frac{1}{2}\right)\left(\frac{\pi}{2}-\frac{2\pi}{2+\pi}\right) - 1 \approx 0.134163549. \tag{22}$$

But, the area above y_2 and below $\sin(x)$ over the interval $[\pi/2, \pi]$ is $1-\frac{\pi}{4} \approx 0.2146018365$, so we have

$$\int_{y=0}^{\pi}(y-\sin(y))^k dy < \int_{y=2\pi/(2+\pi)}^{\pi}((2+\pi)y/\pi-2)^k dy = \frac{\pi^{k+2}}{(2+\pi)(k+1)}. \tag{23}$$

In short, we proved that the error in distance units is $\frac{c}{\rho_n}$, where $\frac{1}{2} < c < \frac{\pi}{2+\pi}$. Hence, the error is inversely proportional to ρ_n. If the sensing range is $R > 1$, and the sensor density (the average number of sensors in the distance unit square) is ρ, it is easy to see that the error is expressed by the formula $\frac{c}{\rho R}$, so the error is inversely proportional to the density and to the sensing range.

To assess the impact of nonsensing nodes for the error, we notice that L measured in radians satisfies $L \geq \frac{1}{2R^2\rho_n} = \frac{1}{2\rho_n}$ and therefore the area over which the nonsensing nodes contributing to error estimate are located is about $R^2(L+\sin(L)) \geq \frac{1}{\rho_n}$. Hence, on average there is one node there. Under assumption of nearly uniform distribution, only j = 1 in Eq. (19) will have nonzero probability, so $P'_1 = 1$ and $P'_{j,j\neq 1} = 0$. Hence, the summation in Eq. (19) collapses to a single element, which is

$$\frac{\int_{t=0}^{L}(t+\sin(t))\,dt}{L+\sin(L)} = \frac{1+L^2/2-\cos(L)}{L+\sin(L)} \approx \frac{L}{2}. \tag{24}$$

From that, we conclude that for medium and large densities (for which L is a fraction, so approximation $\sin(L) \approx L$ is tight), the improvement from using nonsensing nodes tends to 50%.

To verify our accuracy analysis, we also obtained the accuracy from simulation of 800 sensor nodes with sensing range of one unit that were randomly deployed over an area of 20 by 20 units. Hence, $A_d = 400$, $N = 800$. Using Eq. (10), we get that the average accuracy from just nodes that sense the target is $\delta = 0.261$ in radians. Considering $k_{sensing}$ values ranging from 1 to 16, we get from Eq. (20) that the final accuracy with contributions from neighbor sensors that do not sense target is $\epsilon = 0.155$ in radians. On the other hand, the average combination angle from simulation is 0.270, so the accuracy from simulation is $0.270/2 = 0.135$, the difference between the analytical and simulation results is only 0.02 or below 13%, showing an excellent agreement between the two. Another interesting observation is that the use of sensors that do not sense the target in the algorithm improves the accuracy of localization by about 40%, so

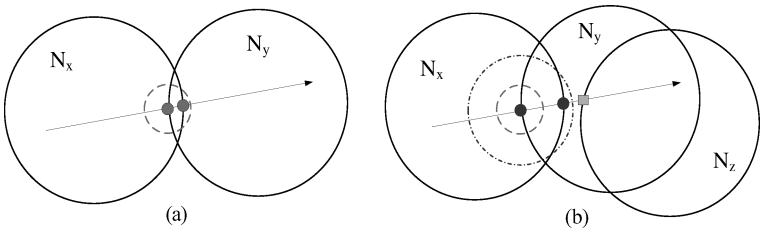


Fig. 12. Illustrations for packet collision and abandoning transmission.

very significantly. Since for this example $\rho_n = 2$ then using the middle of the interval for c we get $L = 0.278$ in radians. Hence, the accuracy predicted by Eq. (24) is 0.139, just within a few percent of the actual accuracy measured in simulation that was 0.135.

5. SIMULATION

5.1 Simulator

We have designed a QT (a cross-platform application framework)-based simulator and used data exchange between multithreads to simulate wireless communication between sensor nodes. The communication range of each sensor node is twice of its sensing range. In the basic simulations for the ideal and imperfect binary sensing model, we assume that there is some MAC (Media Access Control) protocol supporting ideal wireless communication, so simulations have not modeled collisions or dropped packets. To investigate fault-tolerant properties of our tracking algorithm, we added packet collisions to the wireless communication as well as environmental noise to the sensing model for a set of simulations based on the ideal (in terms of detecting the target at exactly the sensing range) binary sensing model.

In our tracking algorithm, a sensor node only broadcasts a message when the target enters or leaves its sensing range, so the only chance that there will be a packet collision is when the target passes over sensing range boundary circles in a quick succession. We assume that the packet collision arises when a passage of the subsequent sensing range boundary happens after the previous one in less than the collision time gap, defined as follows. The collision time gap is the smallest time between the transmissions of two neighboring nodes after which one node will sense the other's transmission and cancel its own broadcast to avoid collision. Otherwise, there will be packet collision and none of the messages broadcast by these two nodes will be received by their joint communication neighbors. A report point is defined as a position of the target on the boundary of the sensing range of a node, which starts transmitting the report of crossing. As shown in Figure 12(a), only if two subsequent report points (circle spots) are within a circle (the dashed line circle) centered at the previous one with the radius of collision_gap*target_velocity, then the corresponding two reports may collide and none of them will be received by the joint communication neighbors of the transmitting nodes. As shown in 12(b), only if only one of two subsequent report points (circle spots) is within the collision circle (the dashed

line circle) but both are within a circle (the dashed dotted line circle) centered at the previous one with radius of transmission_time*target_velocity, the later reporting node may sense the transmission of the previous one and abandon its report.

The environmental noise will also affect the target detection. A sensor node may report wrong target status caused by the false sensing. We randomly select a certain percentage of the nodes in the simulation to report only wrong target status when sensing the target.

5.2 Simulation Setup

It is obvious that the results of our tracking algorithm become more accurate with an increasing number of neighbors reporting the status of the target. To evaluate the impact of the neighbor node count on the performance of target tracking, we used the following simulation environment. We kept the number of nodes fixed at 800 over the area of 800 units by 800 units and varied their sensing range R (R_{out} for the imperfect binary sensing model) from 40 to 150 units. With the same topology, the effect of change of the sensing range is that the average number of (sensing) neighbors increases with the sensing range. Another interpretation is that if we fix the sensing range as a constant, say 40m, then changing the range from 40 to 150 in terms of simulation units corresponds to shrinking the area over which 800 nodes is deployed from 800m by 800m to just 213.3m by 213.3m. In both cases the average number of neighbors of each node changes, but in the second case also the spatial density of sensors per square meter increases. The advantage of this approach is that we can estimate the influence of neighbor node count or sensor spatial density on our tracking algorithm excluding other factors, such as the placement of the sensor nodes in relation to each other, which remains constant in our setting. For the same reasons, the velocity of the target is also adjusted proportionally to the sensing range, making it constant if measured in sensing range units.

For the imperfect binary model, two kinds of detection probabilities are used. The first one is a constant distribution as defined in Eq. (25), where d is the distance between the sensing node and the target.

$$P_{detect1}(d) = \begin{cases} \frac{R_{out}-d}{R_{out}-R_{in}} & R_{in} \le d \le R_{out} \\ 1 & d \le R_{in} \\ 0 & R_{out} \le d \end{cases} \tag{25}$$

The second one is an exponential distribution defined in Eq. (26) [Dhillon and Chakrabarty 2003], where α is its exponent parameter. In order to make the detection probability approximately 0 when $d = R_{out}$, we let $e^{-\alpha(R_{out}-R_{in})} = 0.01\%$, yielding $\alpha = \frac{ln(0.01\%)}{R_{in}-R_{out}}$.

$$P_{detect2}(d) = \begin{cases} e^{-\alpha(d-R_{in})} & R_{in} \le d \le R_{out} \\ 1 & d \le R_{in} \\ 0 & R_{out} \le d \end{cases} \tag{26}$$

Finally, three types of trajectories have been considered, which are linear,

circular, and a piecewise linear with random turns trajectories. To exclude the boundary effect, all the trajectories are confined within the square area with length of $800 - R_{\max}$ located in the middle of the simulated region, where $R_{\max}$ is the maximum sensing range (150 units) in the simulation. For the random turn trajectory, the length of each linear piece of the trajectory is random but proportional to the sensing range. As in Mechitov et al. [2003], we set $R_{\min} = 0.9 \times R_{out}$ under the imperfect binary sensing model.

5.3 Location Estimate

The first metric that we consider is the location estimation error, measured as the ratio of the distance between the estimated and real target locations to the sensing range R (R_{out} for an imperfect binary sensing model).

5.3.1 *Algorithms to be Compared.* We compare our algorithm with the following four other algorithms introduced in Mechitov et al. [2003] and Kim et al. [2005].

(1) *Equal weight*. Target's position is estimated as the average of the detecting sensors' positions.

(2) *Distance weight*. Target's position is estimated as the weighted average of the detecting sensors' positions. The weight for each node is set at $4/\sqrt{4R^2 - v^2t^2}$, where v is the target velocity and t is the time expired since the target has been detected.

(3) *Duration weight*. Target's position is estimated as the weighted average of the detecting sensors' positions. Given the time t that expired since the node has detected the target, the weight of this node is $ln(1 + t)$.

(4) *Line fit*. The initial estimate of the target position is made using distance weight algorithm (2), and then a line that fits the history target position point is found and the current target position is refined using this line and the target velocity.

Algorithms (2), (3), and (4) have been designed for a linear trajectory with constant velocity, so we compare our algorithm with them only for the linear trajectory.

5.3.2 *Simulation Results and Discussion.* We ran each simulation ten times and computed the average and confidence interval of the results under confidence level of 95%. Figure 14 and Figure 15 show the location estimate accuracy results under both ideal and imperfect binary sensing models.

As evidenced by the plots in these figures, in all cases our algorithm's results were better than the results of all four other algorithms. For the ideal sensing case, the ratio of accuracy of our algorithm to the best accuracy of others grows from nearly 3 for an important case of networks with medium neighbor density (sensing range of 40 units) to slightly below 7 for dense networks. For the imperfect sensing case, this ratio is always above 2, with the biggest value of above 3 for networks of medium neighbor density, matching in this case the ratio for ideal sensing. As a result, even for the networks with sensing range $R_{out} = 40$ (i.e., the network in which a node has only five neighbors on average)

and with the imperfect binary sensing model, the algorithm performs very well. Interestingly, for all algorithms there is no significant difference in the results between ideal and imperfect sensing models when the neighbor density is low and medium. However, in networks with higher neighbor node counts, the ideal case yields much better accuracy.

It should also be noted that the location estimate accuracies for all the three trajectory types in the case of our algorithm are nearly the same, showing that our algorithm works well for all kinds of trajectories. There is a slight decrease in accuracy of the algorithm (1), the only one of the comparison algorithms applicable to nonlinear trajectories, for more complicated trajectories.

Another important property of our tracking algorithm is that it achieves the given desired location estimate accuracy using fewer neighbor nodes, that is less dense networks, than needed by the other algorithms discussed before. For example, the accuracy achieved by our tracking algorithm using sensing range of 40 units (so for a network in which a node has 5 neighbors on average) is nearly the same as the accuracy achieved by algorithm (1) using sensing range of 150 units (hence for the network in which a node has 87 neighbors on average) under the ideal binary sensing model. This advantage may be explored in two different ways by deploying the number of nodes higher than necessary for our algorithm but equal to the number of nodes that one of the other algorithms needs to achieve the given location estimate accuracy. First, we can use all the deployed nodes and add certain robustness to node failures to our algorithm. Indeed, if some of the redundant sensors fail (which often happens in the real deployments for a multitude of reasons, such as energy drainage, accidental damage, or random deployment behind obstacles) our algorithm will still provide accurate location estimate results. Second, we can also integrate our tracking algorithm efficiently with the node sleep scheduling mechanism by turning off a portion of neighbor nodes to save energy. In the example mentioned earlier, our tracking algorithm achieves the same location estimate accuracy as algorithm (1) even when up to 95% of neighbor nodes are turned off. This will save a lot of energy and prolong the network lifetime significantly.

We also analyzed the number of messages exchanged and their corresponding energy cost. Let the target move from point O_1 to point O_2 with velocity $v(t)$ over time dt as shown in Figure 13. Let $\alpha = \angle X_1X_2O_2$, then $v(t)dt = 2R\sin(\alpha) \approx 2R\alpha$. Area R_0 contains sensors that will broadcast bit "0" when the target moves from O_1 to O_2, and equal size area R_1 contains sensors that would transmit bit "1". Hence, the total number of messages generated by the the target moving from O_1 to O_2 will be $2A\rho$, where A is the size of area R_0, ρ is the sensor density per unit square. A can be computed from Eq. (8), yielding $A = (2\alpha + \sin(2\alpha))R^2 \approx 4R^2\alpha$. Thus, the total number of messages generated is $4R\rho v(t)dt$. If over time t_r, the target moves distance D, then the number of messages produced is

$$\int_{t=0}^{t_r} 4R\rho v(t)dt = 4R\rho D. \tag{27}$$

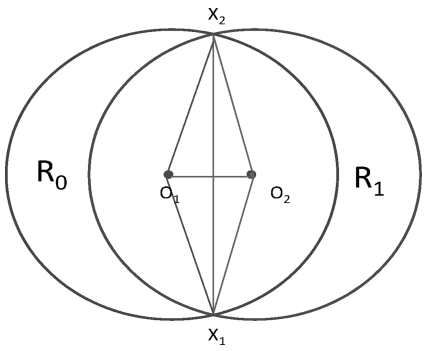


Fig. 13. Analysis of the number of messages generated.

So the total number of messages exchanged is proportional to the sensing range R, the sensor density ρ per unit square, and the distance traveled by the target D.

To verify this analysis, we performed two groups of simulations over an area of size 800 by 800 covered by 800 sensor nodes (so the same density ρ per unit square in each group). We set the sensing range R to be 40 units in one group and 150 units in another. In both groups of simulations, the target moves along a random trajectory (the same for each group) with a constant velocity and over the same distance. We ran the simulation 20 times for each group changing the topology of the network in each run. The exchanged messages were counted. The total number of messages exchanged is 474 when $R = 40$ units and 1759 when $R = 150$ units. The ratio is $1759/474 = 3.71$ which is very close to the ratio of sensing ranges $150/40 = 3.75$.

First, let us consider only transmitting energy cost here. According to the energy consumption model for packet transmission, the energy cost for transmitting a packet to distance d is $E_{tx} = kd^2$, where k is a constant. In the discussed algorithms, each node broadcasts to all its neighbors, so over the distance of the double sensing range, thus $d = 2R$. Hence, the total energy cost is $4R\rho D * k(2R)^2 = 16R^3k\rho D$. From Figure 14, it is clear that our algorithm achieves nearly the same location estimate accuracy using sensing range $R = 40$ units as the algorithm (1) (the best one among all the algorithms compared) using sensing range $R = 150$ units. This means that our algorithm needs to send only $40/150 = 27\%$ messages using only $(40/150)^3 = 2\%$ energy when compared to the algorithm (1).

Considering the receiving energy cost, we noticed that each message is received by $(150/40)^2 = 14$ times fewer nodes in our algorithm than in case of algorithm (1), so the receiving energy expended by our algorithm is only 7% of the receiving energy needed by the algorithm (1).

5.4 Velocity Estimate

We tested the performance of velocity estimation in two scenarios in which the target moves along a linear trajectory. In the first scenario, the target moves at a constant velocity which is $R/15$ unit/second. In the second scenario, the

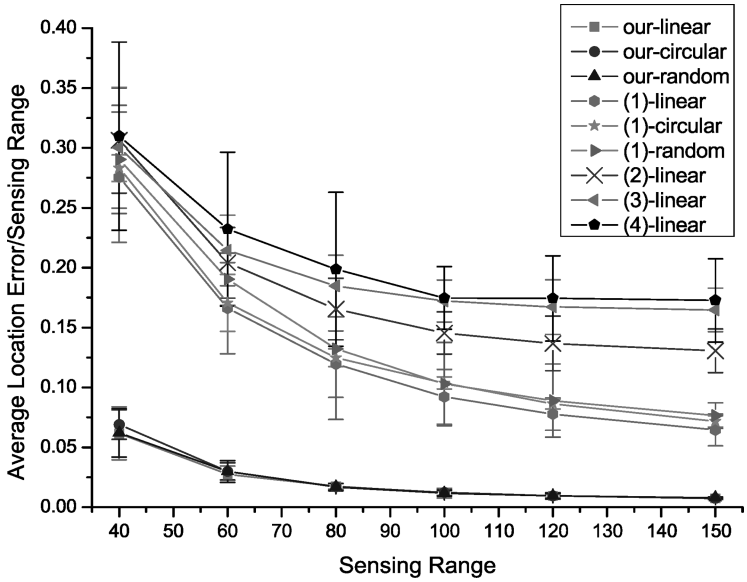
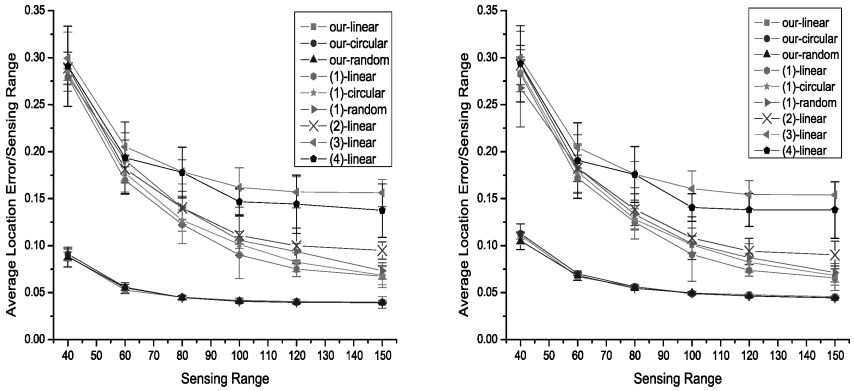


Fig. 14. Location estimate accuracy for ideal binary sensing model.

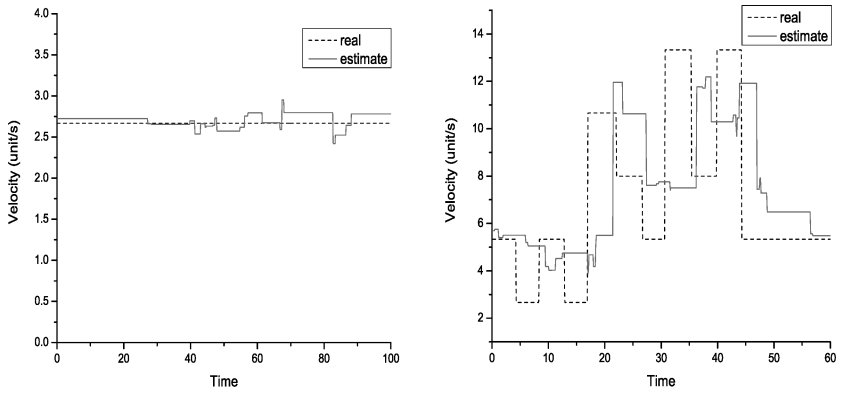


(a) the first detection probability (b) the second detection probability

Fig. 15. Location estimate accuracy for imperfect binary sensing model.

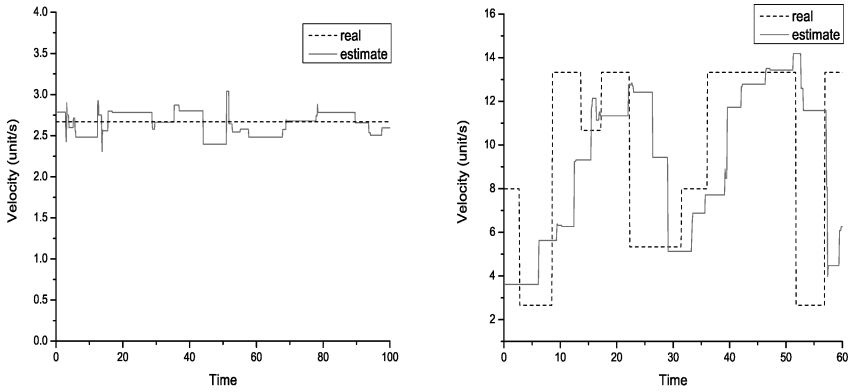
velocity of the target changes suddenly, several times during simulation, to a random value that is a multiple of $R/15$ unit/second. For the ideal binary sensing model, we use the configuration of 800 nodes with $R = 40$ unit (on the average there are five neighbors of each node). For the imperfect binary sensing model, we use the configuration of 800 nodes with $R_{out} = 40$ unit and $R_{in} = 0.9 \times R_{out}$ unit under the first detection probability.

Figure 16 and Figure 17 show the estimated versus real velocities as a function of time in these two scenarios. Clearly, the estimated velocity is very close to the real velocity in the first scenario for both of the binary sensing models. These two also agree well in the second scenario, although there is



(a) constant velocity (b) random velocity

Fig. 16. Velocity estimated for ideal binary sensing model.



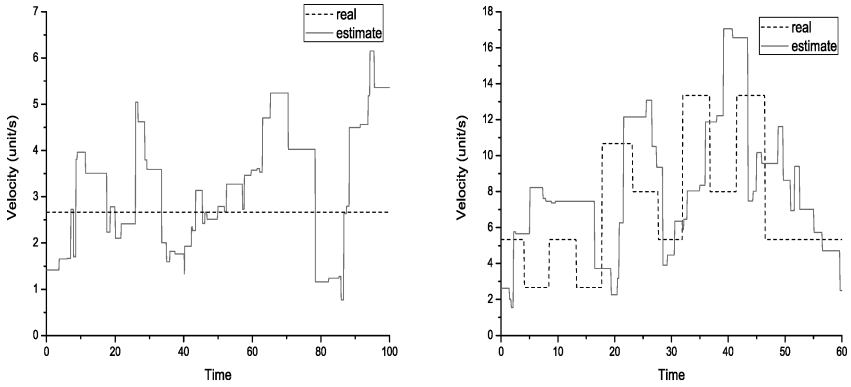
(a) constant velocity (b) random velocity

Fig. 17. Velocity estimated for imperfect binary sensing model.

some delay before the change of real velocity is reflected in its estimate so there are some large deviations in the brief moments immediately after the velocity change.

For the ideal binary sensing model, the average and maximum differences between estimated and real target velocities in the first scenario are 0.0764 unit/second (a few percent of the speed of the target) and 0.2842 unit/second (around 10% of the target's speed). The average and maximum differences between estimated and real target velocity in the second scenario are 2.6295 unit/second and 6.8021 unit/second.

For the imperfect binary sensing model, the average and maximum differences between estimated and real target velocity in the first scenario are 0.1210 unit/second and 0.3713 unit/second, so the average difference nearly doubled, while the maximum difference increased by 30%. The average and maximum differences between estimated and real target velocity in the second scenario are 3.6283 unit/second and 11.5240 unit/second (so nearly 100% of the



(a) constant velocity (b) random velocity

Fig. 18. Velocity estimated using algorithm (1) for ideal binary sensing model.

target's speed). Hence, there is a significant increase in these differences when the sensing model changes from ideal to imperfect.

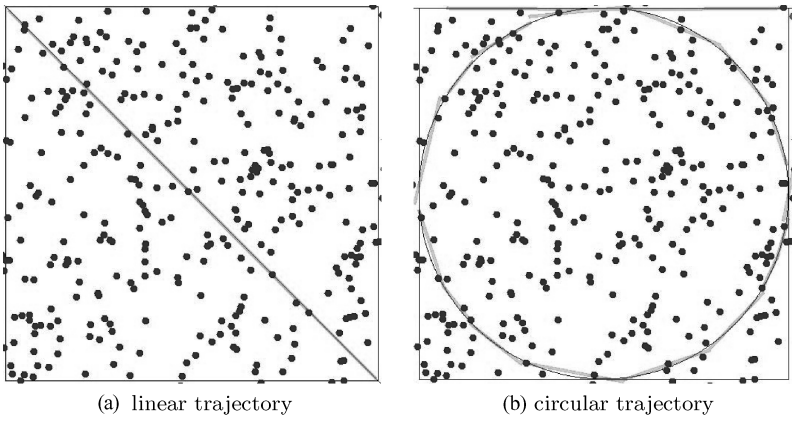
We also performed simulations to obtain velocity estimates using algorithm (1) under the ideal binary sensing model. Figure 18 shows the estimated versus real velocities as a function of time in the same two scenarios. The average and maximum differences between estimated and real target velocity in the first scenario are 1.1689 unit/second (around 44% of the speed of the target) and 3.4833 unit/second (around 130% of the target's speed). The average and maximum differences between estimated and real target velocity in the second scenario are 3.5160 unit/second and 9.0538 unit/second. These differences are much higher than the ones obtained with our algorithm.

5.5 Trajectory Estimate

Figure 19 shows the typical estimations for three trajectory types under the configuration of 800 nodes with $R = 40$ units using the ideal binary sensing model while Figure 20 shows those for the imperfect binary sensing model with $R_{out} = 40$ units and $R_{in} = 0.9 \times R_{out}$ using the first detection probability. The estimation of trajectory uses the method described in Section 3.2.4 with parameters k, n, ϵ set to $k = 30$, $n = 6$, $\epsilon = 0.2$.

We measure the accuracy of estimated trajectory using the average difference between the estimated and real trajectories. It is calculated using the area of a polygon formed by these two trajectories divided by the length of the real target trajectory. The average accuracies are 0.187, 1.227, and 1.704 under an ideal binary sensing model, for linear, circular, and piecewise linear trajectories with random turns, respectively. The average accuracies are 0.287, 1.811, and 2.873 under an imperfect binary sensing model, for linear, circular, and piecewise linear trajectories with random turns, respectively.

We simulated also trajectory estimation using algorithm (1) under the ideal binary sensing model. Figure 21 shows the typical estimations for three trajectories under the same configuration. The average accuracies are 3.926, 4.385, and 5.167, many times higher than the accuracies achieved by our algorithm.



(a) linear trajectory (b) circular trajectory

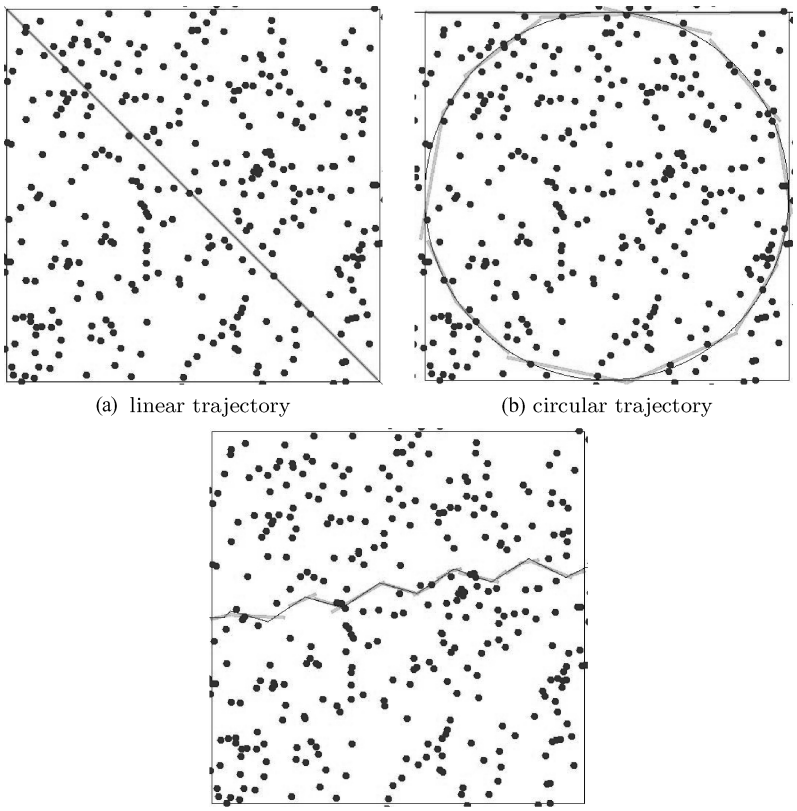
(c) random trajectory

Fig. 19. Trajectory estimation for ideal binary sensing model.

5.6 Fault-Tolerance Simulation

In this section, we evaluate the performance of our algorithm when a sensor occasionally either receives a report with the wrong target status or does not receive a report when the status of the target changes. Such faults arise because of environmental noise distorting sensing or packet collisions disturbing communication with neighbors. We start with the description of modifications to our algorithm that make it tolerant to such faults. Then, we demonstrate the level of fault tolerance achieved by our tracking algorithm through a set of simulations based on the ideal (in terms of detecting the target at exactly sensing range distance) binary sensing model.

To deal with incorrect target status, our tracking algorithm first combines all angles corresponding to "1" bit reports, and then it combines all angles corresponding to "0" bit reports. For each neighbor node that reports "1", the algorithm checks whether its angle intersects with the angles of all the other neighbor nodes reporting "1". For each pair of nodes with angles that do not intersect, a counter of each node will be increased by one and the neighbor node will be added to a list of the partner. Upon completion of this procedure, each node will have its counter and noncommon angle neighbor node lists calculated.



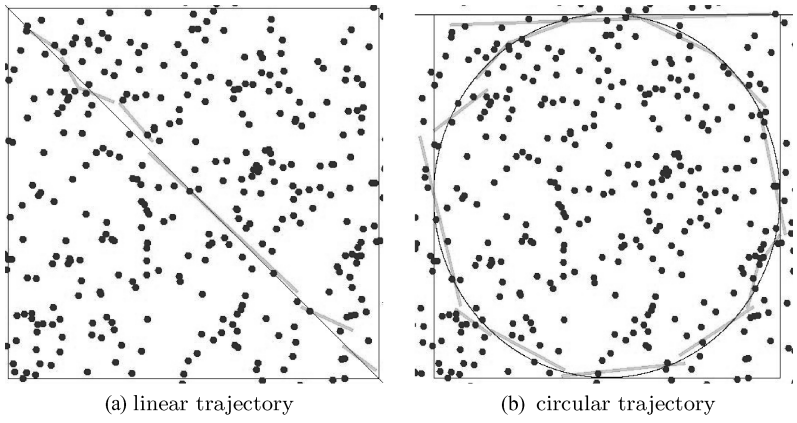
(a) linear trajectory (b) circular trajectory

(c) random trajectory

Fig. 20. Trajectory estimation for imperfect binary sensing model.

For example, the result for the nodes shown in Figure 22(a) is: neighbor node N_w: counter: 2, list: N_y, N_z; neighbor node N_z: counter: 1, list: N_w; neighbor node N_y: counter: 1, list: N_w. Clearly, if all the neighbor nodes report "1" correctly, their counters will be all 0 and their lists will be empty because intersection of all their angles includes the target.

Next, all the neighbor nodes will be put into a list sorted by their counters and the neighbor node with the highest counter will be deleted first. When a neighbor node is deleted, it is also deleted from the noncommon angle neighbor node list of the paired node and the counter for that node will be decreased by one. If multiple nodes have the same and bigger than 0 counter, they are deleted at same time, to make sure that no node with a potentially wrong report survives. This deletion continues until all the counters of the surviving nodes become 0. By then all the error target status reports will be deleted. For example, after deleting node N_w (which has the highest counter, equal to 2, at this point) from the list, we get: neighbor node N_z: counter: 0, list: empty; neighbor node N_y: counter: 0, list: empty. Another example is shown in Figure 22(b): neighbor node N_z: counter: 1, list: N_w; neighbor node N_w: counter: 1, list: N_z; neighbor node N_y: counter: 0, list: empty. After nodes N_w and N_z



(a) linear trajectory (b) circular trajectory

(c) random trajectory

Fig. 21. Trajectory estimation using algorithm (1) for ideal binary sensing model.

with the highest counter equal to 1 are deleted together, the list shrinks to just one node: neighbor node N_y: counter: 0, list: empty. In this example, some correct target status reports are also deleted, but it just affects the accuracy of the estimated target position a little (from the circle spot to the square spot) without ever resulting in a quite far away misleading estimation (the diamond spot, in case that node N_z is considered to report incorrectly alone, causing node N_z count to drop to 0).

After all angles of nodes reporting "1" are combined, the algorithm combines all angles of nodes reporting "0" bit. As shown in Figure 23(a), where $\angle 1o2$ is the angle resulting from combination of all angles of nodes reporting "1" bit, node N_x will find out that there is a contradiction and it will conclude that there is a wrong report included in combining angle $\angle 3o4$ using the "-" operation. For the instance shown in Figure 23(b), where $\angle 3o4$ is caused by a wrong "0" report, the algorithm cannot discover that and the target position will be estimated at the diamond spot. However, if there are enough "1" reports, $\angle 1o2$ will be small, thus the chance that such a situation happens is low. Even if it does happen, the accuracy will not be affected very much because $\angle 1o2$ is small.

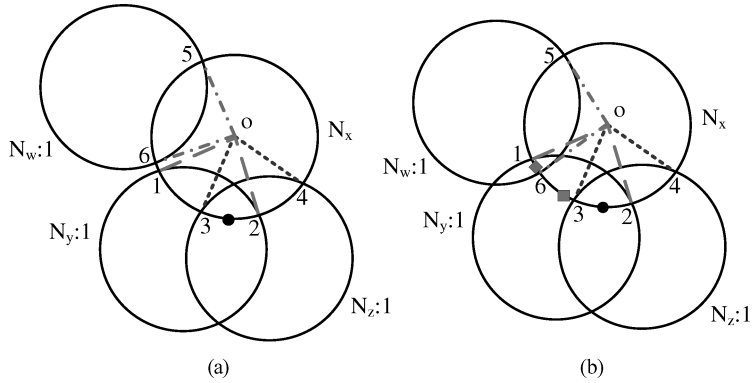


Fig. 22. Examples of fault tolerance in "&" operation.

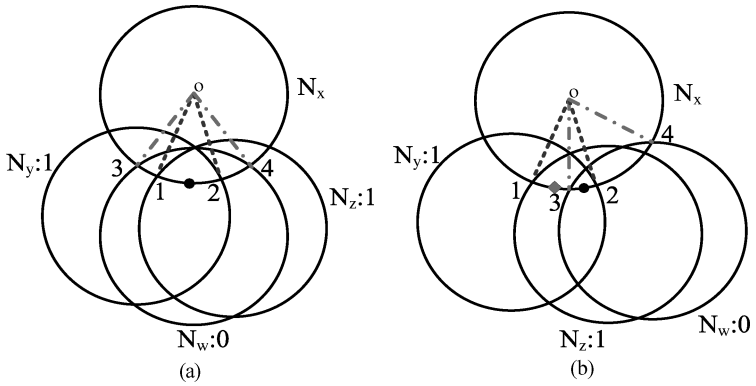


Fig. 23. Examples of fault tolerance in "-" operation.

Another example of fault tolerance is shown in Figure 6(c). If the target falls in angle $\angle a_1 o a_2$, but node N_x receives bit "0" from node N_y due to the detection failure or environmental factors (such as an obstacle preventing target detection), our algorithm can still get the correct target position estimate by excluding node N_y from the neighbor list (described in the previous section) and using information received from the remaining neighbor nodes.

Now, we will give the details of the simulation environment through which we evaluate the fault tolerance of our algorithm. As in the previous simulations, we kept the number of nodes fixed at 800 and varied the sensing range R from 40 to 150 units. A collision gap of the sensor network is defined as the minimum time between two independent attempts to communicate by the neighbor nodes that would not lead to a collision of their transmissions. This gap is determined by the time that it takes for a radio to switch from listening to transmitting. It is usually very small compared to the packet transmission time. We set the collision gap to be 0.0001 second and call the circle of radius 0.0001*target_velocity a collision circle. Thus, the simulation will enforce packet collision if a collision circle centered at the current point of the target crossing the sensing range of a node contains future or past crossing

Table I. Fault-Tolerant Results for Linear Trajectory

Sensing Range	40	60	80	100	120	150
ideal sensing	0.061681	0.027515	0.017458	0.011750	0.009465	0.007501
packet collision	0.061706	0.027889	0.017494	0.012447	0.010685	0.008139
wrong report	0.076377	0.036045	0.021637	0.014421	0.012025	0.009898
collision&wrong	0.077268	0.038050	0.022705	0.015861	0.013133	0.011186

Table II. Fault-Tolerant Results for Circular Trajectory

Sensing Range	40	60	80	100	120	150
ideal sensing	0.068974	0.029909	0.016972	0.012357	0.009378	0.007449
packet collision	0.069416	0.030191	0.017090	0.012962	0.010335	0.008778
wrong report	0.080432	0.039182	0.022370	0.015434	0.012599	0.009409
collision&wrong	0.080896	0.037735	0.023686	0.016244	0.012954	0.011477

Table III. Fault-Tolerant Results for Random Trajectory

Sensing Range	40	60	80	100	120	150
ideal sensing	0.062041	0.029779	0.016823	0.011575	0.009336	0.007970
packet collision	0.064382	0.032705	0.017623	0.012588	0.010028	0.008748
wrong report	0.074263	0.034594	0.022625	0.015810	0.012005	0.010031
collision&wrong	0.080263	0.035148	0.023778	0.016603	0.012753	0.011027

points (see Figure 12). It should be noted that the future crossing points are known only in simulation. Moreover, the preceding condition is necessary but not sufficient (for example, for nonlinear trajectories the predicted collisions may not happen when the target changes the direction of movements), so this collision enforcing method overestimates the number of collisions. We also set the transmission time to be 0.01 second. In our algorithm, if a node wants to transmit a report and overhears that another node is already transmitting, it will abandon the transmission and would not report the change of status. This approach decreases accuracy of the method but avoids cascading collisions, when the delayed node transmission would collide with the later transmission by another node. The simulator uses the transmission circle defined as a circle of radius 0.01*target_velocity to detect such situations. Indeed, it can arise only when the transmission circle centered at the current point of the target crossing a sensing range of a node contains any previous crossing points (like in case of collisions, this method overestimates the number of abandoned reports).

We set the probability that the sensor node reports wrong target status “0” even if the target is within its sensing range to be 5% and set the probability that the sensor node reports wrong target status “1” even if the target is out of its sensing range to be 1%. These settings reflect the fact that the chance for the wrong report of the first kind (that is, failing to detect that the target is within the range of the sensor) is higher than that of the second kind (that is, detecting a “phantom” target in the sensing range).

Tables I to III show the target position estimation accuracy for linear, circular, and random trajectories under the ideal binary sensing model, packet collision model, wrong report model, as well as packet collision together with wrong report model, respectively. Clearly, the accuracy decreases only a little when the packet collision and wrong target status report are taken into

consideration, which means that our tracking algorithm has a good fault-tolerant property.

6. CONCLUSIONS AND FUTURE WORK

Target tracking is a typical and important application of sensor networks usually relying on cooperation between sensor nodes. In this article, we study the target tracking problem under the simple and basic binary sensor network model. We introduce a real-time distributed target tracking algorithm without time synchronization for both the ideal and imperfect binary sensing models which is also energy efficient and fault tolerant. We analyze the accuracy of our algorithm under the ideal binary sensing model and demonstrate that it agrees well with the accuracy obtained via simulations. The analysis also shows that for the configuration simulated, the use of sensors that do not sense the target by the algorithm improves the accuracy of localization by nearly of factor of 2 (decreasing the estimation error by 50% compared to using only sensors that do sense the target), so very significantly. Results of extensive simulations of this algorithm performed under different configurations and scenarios are also reported and they confirmed the analysis. We observe that the introduced algorithm outperforms algorithms reported elsewhere in terms of its accuracy of estimating the target location, velocity, and trajectory using the binary sensor networks.

Our future work will further investigate energy efficiency in target tracking applications. Target tracking systems using sensor networks spend most of the energy on sensing and communicating measurements between sensors. Since sleeping is the most basic and effective way to conserve energy, nontracking sensors should sleep, while tracking sensors should spend the minimum energy necessary to track the mobile target. A sleep scheduling mechanism through which unnecessary sensor nodes can be turned off and go to sleep effectively enhances the lifetime of the entire system [Yang et al. 2006; Xu et al. 2004; Yeow et al. 2007; Visvanathan and Veeravalli 2005]. Therefore, integrating our algorithm with a sleep scheduling protocol to reduce the energy consumption in the target tracking applications will be the main subject of our future work.

REFERENCES

ARORA, A., DUTTA, P., BAPAT, S., KULATHUMANI, V., ZHANG, H., NAIK, V., MITTAL, V., CAO, H., DEMIRBAS, M., GOUDA, M., CHOI, Y.-R., HERMAN, T., KULKARNI, S. S., ARUMUGAM, U., NESTERENKO, M., VORA, A., AND MIYASHITA, M. 2004. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Int. J. Comput. Telecomm. Netw. 46*, 5, 605–634.

ASLAM, J., BUTLER, Z., CONSTANTIN, F., CRESPI, V., CYBENKO, G., AND RUS, D. 2003. Tracking a moving object with a binary sensor network. In *Proceedings of the ACM SIGOPS International Conference on Embedded Networked Sensor Systems (SenSys)*.

BROOKS, R. R., RAMANATHAN, P., AND SAYEED, A. 2003. Distributed target classification and tracking in sensor networks. *Proc. IEEE 91*, 8, 1247–1256.

CHONG, C.-Y. AND KUMAR, S. P. 2003. Sensor networks: evolution, opportunities, and challenges. *Proc. IEEE 91*, 9, 1247–1256.

DHILLON, S. AND CHAKRABARTY, K. 2003. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*. Vol. 3. 1609–1614.

DJURIC, P. M., VEMULA, M., AND BUGALLO, M. F. 2004. Signal processing by particle filtering for binary sensor networks. In *Proceedings of the 11th IEEE Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop*. 263–267.

FEDDEMA, T. J., BARRY, L., AND SPLETZER. 1999. Probability of detection for cooperative sensor networks. In *Proceedings of the SPIE Unattended Ground Sensor Technologies and Applications*. Vol. 3713. 1–11.

GENTILE, C. 2007. Distributed sensor location through linear programming with triangle inequality constraints. *IEEE Trans. Wirel. Comm*. Vol. *6*. 2572–2581.

JING, T., HICHEM, S., AND CEDRIC, R. 2007. Binary variational filtering for target tracking in sensor networks. In *Proceedings of the 14th IEEE/SP Workshop on Statistical Signal Processing*. 685–689.

KIM, W., MECHITOV, K., CHOI, J.-Y., AND HAM, S. 2005. On target tracking with binary proximity sensors. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*.

LI, D., WONG, K., HU, Y. H., AND SAYEED, A. 2002. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Process, Mag. 19*, 2.

LIN, C.-Y., PENG, W.-C., AND TSENG, Y.-C. 2006. Efficient in-network moving object tracking in wireless sensor networks. *IEEE Trans. Mobile Comput. 5*, 8, 1044–1056.

LIU, X., ZHAO, G., AND MA, X. 2007. Target localization and tracking in noisy binary sensor networks with known spatial topology. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vol. 2. II–1029–II–1032.

MECHITOV, K., SUNDRESH, S., KWON, Y., AND AGHA, G. 2003. Cooperative tracking with binary-detection sensor networks. Tech. rep. UIUCDCS-R-2003-2379, University of Illinois at Urbana-Champaign.

RAHMAN, R., ALANYALI, M., AND SALIGRAMA, V. 2007. Distributed tracking in multihop sensor networks with communication delays. *IEEE Trans. Signal Process. 55*, 9, 4656–4668.

SAVVIDES, A., HAN, C.-C., AND STRIVASTAVA, M. B. 2001. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*. 166–179.

SHRIVASTAVA, N., MUDUMBAI, R., MADHOW, U., AND SURI, S. 2006. Target tracking with binary proximity sensors: Fundamental limits, minimal descriptions, and algorithms. In *Proceedings of the ACM SIGOPS International Conference on Embedded Networked Sensor Systems (SenSys)*.

SINGH, J., MADHOW, U., KUMAR, R., SURI, S., AND CAGLEY, R. 2007. Tracking multiple targets using binary proximity sensors. In *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks*. 529–538.

VISVANATHAN, A. AND VEERAVALLI, V. V. 2005. Sleeping policies for energy-efficient tracking in sensor networks. In *Proceedings of the IEEE/SP 13th Workshop on Statistical Signal Processing*. 1158–1163.

WANG, Z., BULUT, E., AND SZYMANSKI, B. K. 2008a. A distributed cooperative target tracking with binary sensor networks. In *Proceedings of the IEEE International Conference on Communication (ICC) Workshops*. 306–310.

WANG, Z., BULUT, E., AND SZYMANSKI, B. K. 2008b. Distributed target tracking with imperfect binary sensor networks. In *Proceedings of the IEEE Global Telecommunications Conference (Globecom), Ad Hoc, Sensor and Mesh Networking Symposium*. 1–5.

XU, Y., J., W., AND LEE, W.-C. 2004. Prediction-based strategies for energy saving in object tracking sensor networks. In *Proceedings of the IEEE International Conference on Mobile Data Management*. 346–357.

YANG, L., FENG, C., AND PENG, R. J. J. 2006. Binary variational filtering for target tracking in sensor networks. In *Proceedings of the IEEE International Conference on Networking, Sensing and Control*. 916–921.

YEOW, W.-L., THAM, C.-K., AND WONG, W.-C. 2007. Energy efficient multiple target tracking in wireless sensor networks. *IEEE Trans. Vehicular Technol. 56*, 2, 918–928.

ZHAO, F., SHIN, J., AND REICH, J. 2002. Information-Driven dynamic sensor collaboration for tracking applications. *IEEE Signal Process. Mag. 19*, 2, 61–72.

Received June 2008; revised October 2009; accepted October 2009