A Hop-Oriented Position Estimation Algorithm for Wireless Sensor Networks

Hsing-Lung Chen, Li-Chung Lin
Department of Electronic Engineering
National Taiwan Univ. of Sci. and Tech.
Taipei, Taiwan
hlchen@mail.ntust.edu.tw, M9302111@mail.ntust.edu.tw

Shu-Hua Hu
Department of Comp. Sci. and Info. Eng.
Jinwen Univ. of Sci. and Tech.
New Taipei, Taiwan
shhu@just.edu.tw

***Abstract*—A wireless sensor network consists of a group of sensor nodes that broadcast the sensed data to the base station hop by hop via radio frequency. It is useful only if the sensed data are associated with the locations of the sensor nodes. Therefore, location estimation of sensor nodes has become an important issue. Ideally, each sensor node can obtain its location by being equipped with a GPS device. However, this approach costs too much, contradicting the objective of low cost sensor nodes. Hence, it is reasonable that a few sensor nodes only are equipped with a GPS device, and the others estimate their own locations by way of the collected information. The purpose of this paper is to propose a hop-oriented position estimation algorithm (HOPE). Four beacon nodes, each equipped with GPS, broadcast the hop count information, and normal nodes rebroadcast the information after receiving it. Finally, normal nodes employ the received hop count information to estimate their own locations with simple calculations. The simulation results show that the proposed algorithm has better accuracy for location estimation than other proposed methods.**

Keywords- sensor nodes; GPS; broadcast; hop count; beacon nodes; normal nodes

I. INTRODUCTION

Recent advances in MEMs (MicroElectroMechanical Systems) technology and wireless communications technology have provided micro-electronic devices with the capabilities of accurate sensing, communication and computation; such devices are called sensor nodes [1]. Because the sensor node hardware is designed with the objectives of low cost, small size and low power consumption, there are great restrictions on sensor nodes' power, memory and computation ability.

A wireless sensor network (WSN) consists of a group of sensor nodes that broadcast the sensed data to the base station hop by hop via radio frequency. These sensor nodes can be randomly deployed in a region by the hundreds or thousands. However, such a network is useful only if the sensed data are associated with the locations of the sensor nodes. Therefore, how to get the precise location information with a reasonable amount of effort is an important issue in WSN design.

In WSNs, some research assumes that sensor nodes maintain their own location information. This can be realized if the WSNs are situated in a small region, or if the sensor nodes can be placed regularly. In recent years, because of the development of the Global Positioning System (GPS) [2], it is assumed that all sensor nodes can obtain location information by GPS. However, the price of GPS is more expensive than the sensor node itself. Hence, it is unreasonable that each sensor node be equipped with GPS. A more reasonable approach is to assume that a few sensor nodes (named beacon nodes) are equipped with GPS, and the others (named normal nodes) estimate their own locations by way of the collected information. This paper introduces a hop-oriented position estimation algorithm (HOPE). HOPE only employs four beacon nodes to broadcast the hop count information. Normal nodes can estimate their own locations more accurately with simple calculations by way of the collected hop count information.

This paper is organized as follows. In Section 2, we introduce some of the related works in location estimation research. Section 3 describes our Hop-Oriented Position Estimation algorithm. Section 4 presents the simulation results. Section 5 concludes this paper.

II. RELATED WORK

Current location estimation algorithms for WSNs can generally be categorized as range-based and range-free approaches.

The range-based approaches estimate the locations of sensor nodes by the information of distance and angle between two different sensor nodes. For example, the Time of Arrival (TOA) approach measures the relative distance between two nodes by the signal propagation time [3]. However, it is hard to synchronize the time of all sensor nodes, resulting in poor accuracy of estimation. Another approach, Time Difference of Arrival (TDOA) [4], [5], [6], is similar to TOA. However, for TDOA, the sensor nodes are equipped with both ultrasound and RF hardware to solve the problem of time synchronization, resulting in good accuracy of estimation. Angle of Arrival (AOA) employs directional antennae or digital compasses [7] to measure the relative angle between two sensor nodes. The Received Signal Strength Indicator (RSSI) [4], [8], [9], [10], [11], [12] approach translates the received signal strength to the estimated distance between the receiving node and the transmitting node based on the radio propagation theory. Nevertheless, the received signal strength is often affected by the physical-layer problems inherent in RF systems, such as multi-path fading, unstable signal propagation and background noise. The range-based approaches often need additional equipment to improve the accuracy of location estimation. Hence, these approaches are not suitable for resource-limited WSNs.

Because the cost of range-based approaches is higher, many range-free approaches are proposed. For example, DV-Hop [7], [13] is based on the concept of distance vector routing (DV-routing). DV-Hop refers to location-aware nodes as anchors. Each anchor broadcasts its position throughout the network. All nodes should maintain the minimum hop count to each anchor in the table. Each anchor can calculate the average distance per hop by hop count and the distance, and then broadcast it to the neighboring normal nodes. Normal nodes can estimate their location based on the minimum hop count table, the average distance per hop, and the anchors' locations. The GRIPHON [14] scheme also utilizes hop count information for location estimation. Four nodes with known locations called markers are placed at the four corners of the network region. Like anchors, markers broadcast their positions to their neighbors. The other nodes can obtain the minimum hop count of each marker. The hop count vector (*hcv*) of node *k* is represented as $H_k = (h_1, h_2, ..., h_i)$, where h_i is the minimum hop count from node *k* to marker *i*. The network region is further subdivided into small grid zones. Then the control center has already calculated a mean *hcv* for each grid zone. At the control center, the *hcv* of each node is compared with the mean *hcvs* of all grid zones to determine in which grid zone the node is residing. The sensor node's location is estimated as the central point of its residing grid zone. The approaches employing hop count information to do location estimation require a great deal of communication, and only work well in dense networks.

The Convex Position Estimation (CPE) [15] algorithm assumes that if a normal node can receive the broadcasts of neighboring beacon nodes, it must reside in the overlapping region of the communication range of these beacon nodes. CPE defines the estimative rectangle (ER) as the smallest rectangle covering the overlapping convex, and regards the center of the rectangle as the estimative position of the normal node. However, it needs a great deal of computation to obtain the overlapping convex. Hence, CPE needs a central controller to estimate the position of each normal node and to flood the estimative position back to each normal node. Moreover, CPE has poor scalability due to the heavy traffic load.

Like CPE, the Distributed Location Estimation (DLE) [16] algorithm employs the ER to estimate locations of normal nodes. However, DLE simplifies the computation by replacing complicated functions with simple operations. Therefore, the computation can be distributed to every normal node, instead of just being performed by the central controller. In order to improve the accuracy of the estimative location, DLE introduces the concept of the farther neighboring beacon nodes, whose communication range may overlap the ER and does not cover the normal node. Then normal nodes can adjust their ERs by excluding the communication range of the farther neighboring beacon node, from the original ER. Thus, the accuracy of location estimation can be improved.

III. HOP-ORIENTED POSITION ESTIMATION ALGORITHM

The preliminary version of this paper is in [17]. In this section, we describe the HOPE algorithm for location estimation in WSNs. Our HOPE algorithm makes the following assumptions:

- Each sensor node has a unique ID.
- Sensor nodes are deployed randomly.
- Beacon nodes obtain their location by GPS.
- Normal nodes need to estimate their own locations.
- Beacon nodes have six kinds of communication range (1β, 2β, 3β, 4β, 5β and 6β), where β is the unit of one hop.
- Normal nodes have three kinds of communication range (4β, 5β and 6β).

Our proposed HOPE algorithm not only achieves more accurate location estimation but also employs less beacon nodes than other approaches. There are two phases in the HOPE algorithm. We will describe these in the following section.

A. *Data Collection Phase*

In the HOPE algorithm, each normal node estimates its own location by the number of hop counts to every beacon node. To make the estimation more accurate, we assume that one beacon node is placed in each of the four corner areas at random, as shown in Fig. 1.

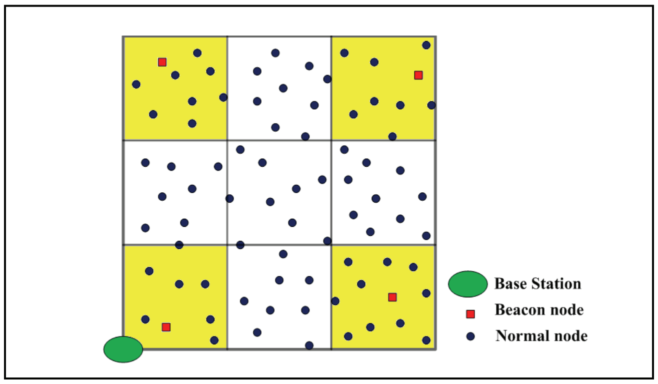

Figure 1. Example of network region.

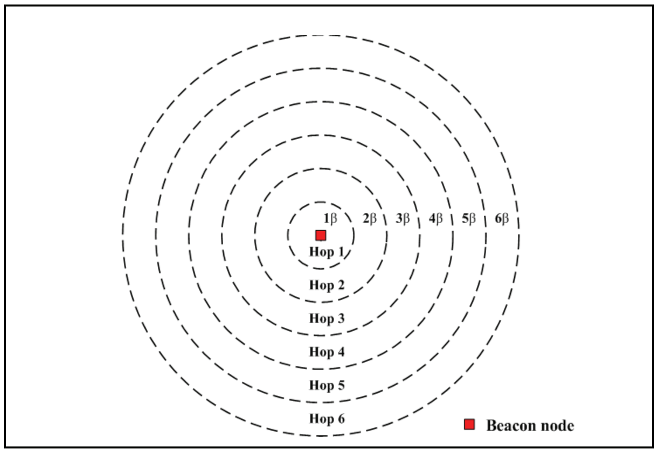

Figure 2. The scenario of beacon node broadcasting.

Because it is difficult to synchronize the time of all sensor nodes, HOPE lets the base station be the master to control the sensor nodes. The base station will notify a beacon node that it can start broadcasting hop count information. After a fixed time interval, the base station will notify the next beacon node to start broadcasting. For instance, beacon *A* first broadcasts information with hop count 1 to its neighbors by using the transmission range 1β. Each node, when it receives the hop count information 1, will record the hop count 1. Then beacon *A* will change the transmission range from 2β to 6β successively and broadcast information with hop counts from 2 to 6, respectively. Rings 1 to 6 are formed. Each node records the coordinates of beacon *A* and the smallest hop count to beacon *A*. The scenario of beacon node broadcasting is shown in Fig. 2.

In order to reduce broadcasting collisions, the concept of a "broadcast slot" is proposed. We let nodes start forwarding the hop count information according to their hop counts. The beacon node will broadcast the information with hop counts from 1 to 6 in slot 0. The nodes with received hop count 1 will broadcast the information with hop counts from 5 (1+4) to 7 (1+6) by using the transmission range from 4β to 6β, respectively, in slot 1. Nodes with the received hop count *k* will broadcast the information with hop counts from (*k*+4) to (*k*+6) in slot *k*. Each node will keep the smallest hop count to the beacon. The nodes in ring *j* will receive the messages with hop count *j* from the nodes in rings *j*-4, *j*-5 and *j*-6. This redundancy can reduce the effect of message collision and non-uniform distribution of sensor nodes. An example of broadcast slots is shown in Fig. 3. The maximum number of slots can be estimated with the regional size and β.

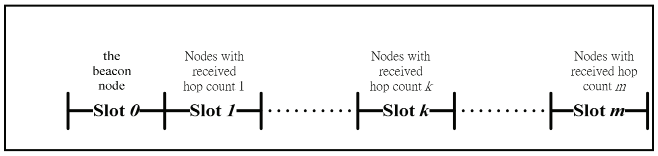
the beacon node
Nodes with received hop count 1
Nodes with received hop count *k*
Nodes with received hop count *m*
Slot *0*
Slot *1*
Slot *k*
Slot *m*

Figure 3. An example of broadcast slots.

The broadcast data contains the hop counts of this message and sender, the coordinates of the beacon node, and the remaining time. The remaining time is defined as the time interval between the sending time and the ending of the current slot. When a node receives the broadcast data, it can calculate the waiting time of its broadcasting slot, which is equal to the remaining time plus the remaining slots. Then the node starts broadcasting information randomly in its broadcasting slot. We use relative time to solve the problem of time synchronization. Nodes will broadcast data in their corresponding broadcasting slots at random to reduce the occurrence of broadcasting collisions.

After four beacon nodes (*A, B, C, D*) finish broadcasting, each node will have the smallest hop counts away from four beacon nodes, (*h(A), h(B), h(C), h(D)*), called hop count vector (*hcv*).

The nodes within the same ring will have the same received hop count. Considering the coverage of a beacon node's broadcast, it will form six concentric circles. We hope that further coverage of broadcastings approach the concentric circles. Therefore, other nodes broadcast information using three kinds of transmission range (4β, 5β, 6β). After the concentric circles for the four beacon nodes are overlaid, the region will be divided into several disjointed areas, called vector blocks. Nodes in the same vector block have the same *hcv*. Fig. 4 shows an ideal vector block situation.

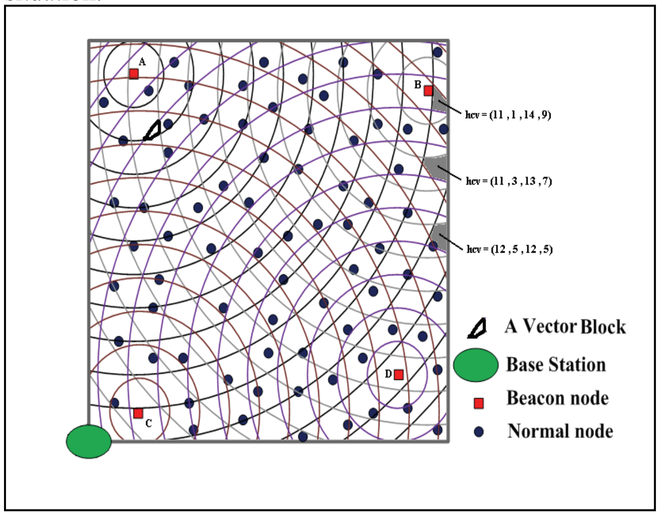

Figure 4. An ideal vector block situation.

B. *Position Estimation Phase*

After each node has obtained a hop count vector, the base station will broadcast the message of location estimation. Each node can estimate its own location by using the *hcv* and the coordinates of the beacon nodes. Each vector block, which is formed with the overlapping area of four corresponding rings, has a unique *hcv*. Each node estimates its location as follows:

1. A normal node first chooses two beacon nodes. The intersection of their corresponding rings will be two quasi-parallelograms. The centers of the two quasi-parallelograms can be derived by intersecting the central lines of two corresponding rings. One quasi-parallelogram closely related to the hop count vector is selected as the target quasi-parallelogram.
2. Then it chooses one of the two remaining beacon nodes. The intersection region will be formed by overlapping this corresponding ring and the target quasi-parallelogram. In the center line of this corresponding ring, find a point closest to the center of the target quasi-parallelogram. The middle point between this point and the center of the target quasi-parallelogram is regarded as the center of this intersection region.
3. Repeat the last procedure for the last beacon node. The center of the vector block is regarded as the estimated location of this normal node.

C. *Quasi-parallelograms*

During the process of finding the target quasi-parallelogram, several problems occur. If we choose any two beacon nodes, the intersection of the two corresponding rings may not be two disjointed quasi-parallelograms. It is difficult to calculate the center of the target quasi-parallelogram. The angle with respect to two selected beacon nodes is defined as the angle from the normal node to the two selected beacon nodes, which can be estimated by the hop count vector. The two beacon nodes whose corresponding angles are closest to a right angle are selected, because the corresponding quasi-parallelogram is the closest to a rectangle.

Another problem is that a normal node will select an incorrect target quasi-parallelogram. If one of two selected beacon nodes is closest to the normal node, we may select an incorrect target quasi-parallelogram because two quasi-parallelograms are too close to distinguish by inspecting the hop count vector. In selecting two beacon nodes, that which is closest to the normal node is not considered. Consider the example in Fig. 5. Beacon node *A* is not considered because it is closest to the normal node. There are three combinations left to be considered: *BC*, *BD* and *CD*. Beacon nodes *B* and *C* are selected because the angle with respect to beacon nodes *B* and *C* is closest to a right angle. The quasi-parallelogram close to beacon node *A* is selected as the target.

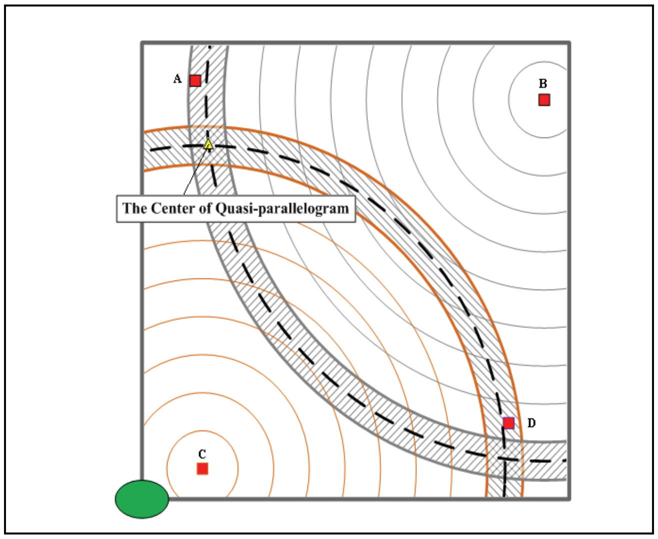

Figure 5. An example of obtaining the center of the target quasi-parallelogram.

D. *Intersection Region*

Recall that a vector block is the intersection of four corresponding rings. The target point of the central line of a ring is defined as a point on the central line closest to the center of the target quasi-parallelogram. At first, we choose one beacon node with the shorter distance from its corresponding target point to the target quasi-parallelogram. Thus, we can obtain a bigger intersection region by intersecting the target quasi-parallelogram and the corresponding ring.

The center of the intersection region can be estimated as the middle point between the center of the target quasi-parallelogram and the corresponding target point, as shown in Fig. 6. Finally, the vector block is the intersection of the intersection region and the last corresponding ring. The center of the vector block can be estimated as the middle point between the center of the intersection region and the corresponding target point, as shown in Fig. 7.

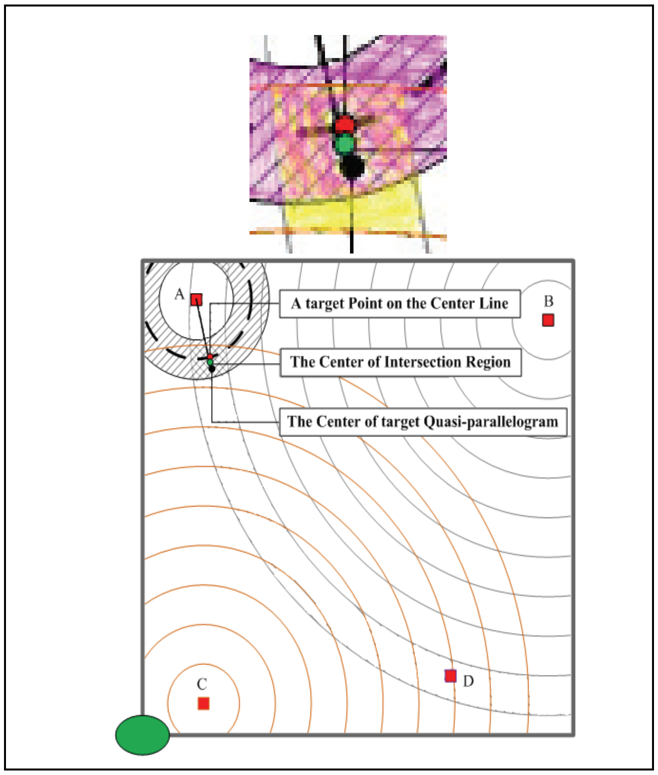
A
B
A target Point on the Center Line
The Center of Intersection Region
The Center of target Quasi-parallelogram
C
D

Figure 6. An example of obtaining the center of the intersection region.

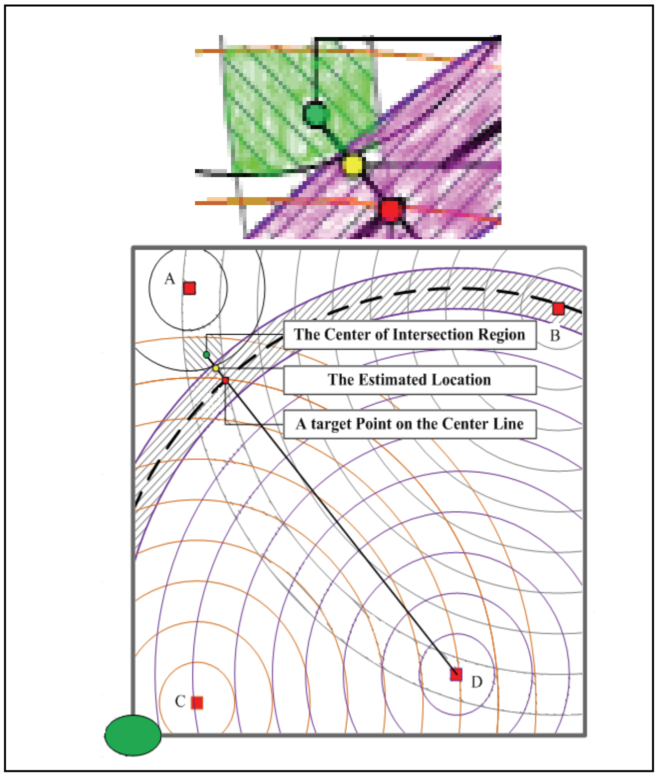
A
B
The Center of Intersection Region
The Estimated Location
A target Point on the Center Line
C
D

Figure 7. An example of obtaining the estimated location.

IV. Simulation and Results

In order to demonstrate the accuracy and feasibility of the HOPE algorithm, we will conduct the simulations in two kinds of situations.

At first, we compare the Distributed Location Estimation algorithm with HOPE. We assume that there are no collisions, and energy consumption is not considered. The simulation area is a square region of side length 1,000m. The total number of nodes is tuned between 50 and 250. To show the accuracy, we compute the mean error which is the average distance between the estimated and actual locations. Then HOPE uses Network Simulator version 2 (NS-2) to further consider the collisions. The 802.11 MAC protocol with the proposed broadcast slot mechanism is employed. To show the feasibility, the mean errors of the direct computing version and the NS-2 version are compared.

A. *Impact of the Density of Sensor Nodes*

Because the DLE algorithm needs a proportion of beacon nodes, the ratio of beacon nodes to the total nodes is set to 40%. Fig. 8 shows the impact of the density of sensor nodes on the mean error. When the node density is low (less than 150 nodes), DLE has poor performance. Although DLE has good performance when the node density is high, the number of beacon nodes will be increased. Under any node density, HOPE has a smaller mean error than DLE. To show the stability of both algorithms, we define the mean error range of DLE and HOPE. Fig. 9 shows that the mean error range of HOPE is smaller than that of DLE. Therefore, it is shown that our HOPE algorithm is more stable than the DLE algorithm.

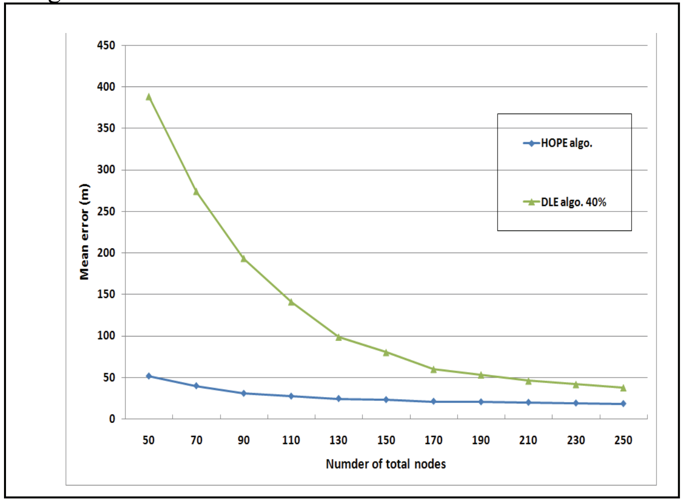

Figure 8. Node density vs. mean error.

B. *Impact of the Time Synchronization and Collisions*

In HOPE, each node obtains needed information by way of broadcasting messages. Collisions will happen in actual networks. In order to prove the feasibility of the algorithm, we conduct the simulations in NS-2, and compare it with the collision-free version. Fig. 10 shows the comparison of the collision-free version and the NS-2 version in terms of mean error. In low node density, the NS-2 version has more errors than the collision-free version because some nodes cannot receive the correct hop count information due to collisions. In high node density, the NS-2 version has almost the same mean error as the collision-free version, because most of the nodes can receive the correct hop count information from at least one node without collisions. It is shown that the broadcast slot mechanism can avoid most collisions successfully.

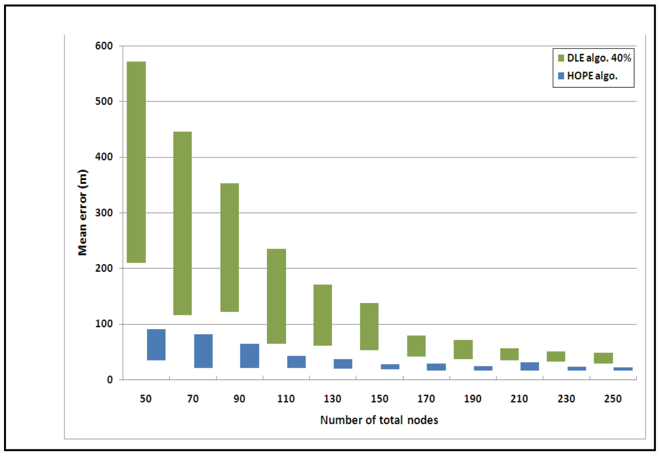

Figure 9. The experiment result of mean error range.

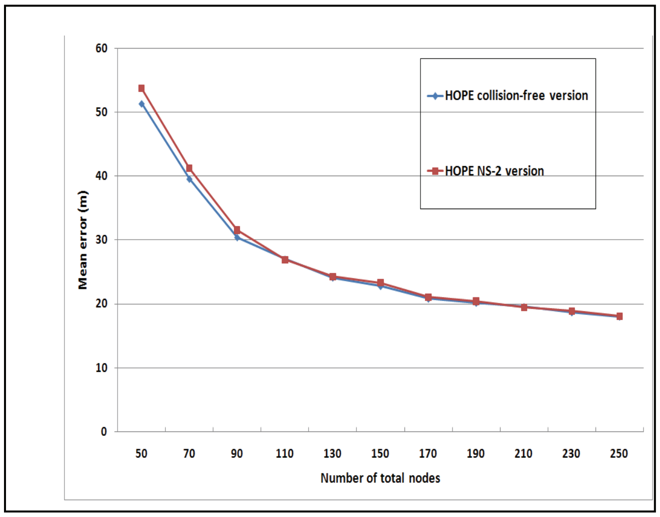

Figure 10. The comparison of the collision-free version and the NS-2 version.

C. *Enhancing the Accuracy of the Algorithm*

Based on the concept of vector blocks, it could be inferred that if the vector block can become smaller, normal nodes in the vector block will be closer to the center of the vector block, resulting in a decrease in the mean error. The vector block can be diminished by decreasing the β value. Fig. 11 shows the mean error in low node density by tuning the β value between 20 and 50, and using a 95% confidence interval. From this figure, it is found that if node density is less than some value with respect to β, mean error is increased abruptly, because some nodes cannot receive the correct hop count information.

Fig. 12 shows the mean error in high node density and using a 95% confidence interval. It is found that if the node density is sufficient, the mean error will be smaller by reducing the β value.

From Figs. 11 and 12, we know that when the density is less than 400 (node/km^2), the use of $\beta = 50$ is better. When the density is between 400 and 700, we should use $\beta = 40$. When the density is between 700 and 1,000, we can employ $\beta = 30$. When the density is from 1,000 to 5,000, we can use $\beta = 20$. If the density is higher than 5,000, we recommend the use of $\beta = 10$. Users can thus adjust the β value to obtain a better mean error.

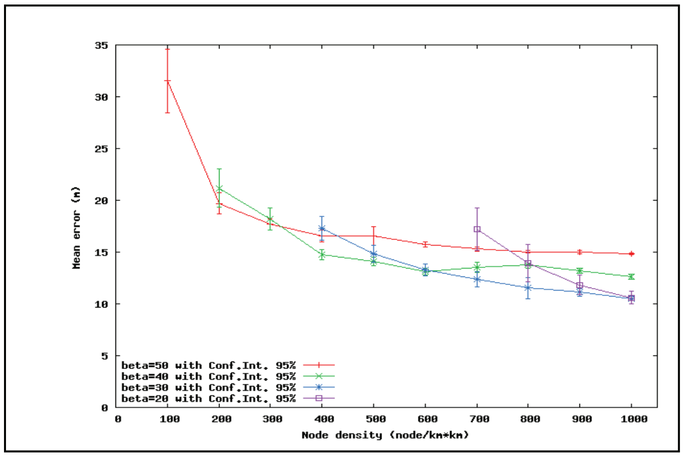

Figure 11. Adjusting β value in low node density.

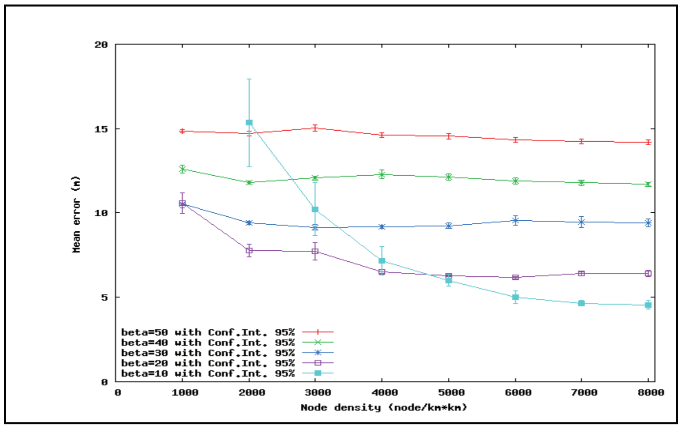

Figure 12. Adjusting β value in high node density.

V. CONCLUSIONS

WSNs are useful only if the sensed data are associated with the locations of the sensor nodes. It is reasonable that a few sensor nodes are equipped with a GPS device and that the others estimate their own locations by way of collected information. In order to reduce costs while maintaining estimation accuracy, the HOPE algorithm is proposed.

The simulation results show that the HOPE algorithm not only uses fewer beacon nodes, but also has higher accuracy as a result of tuning the β value. In simulations of the NS-2 version, it is found that a broadcast slot mechanism can reduce collisions significantly, resulting in high stability of the HOPE algorithm. Implementing the HOPE algorithm in real *WSNs* is worthy of further investigation.

ACKNOWLEDGMENT

This work was supported, in part, by the National Science Council, Republic of China, under Grant No. 99-2218-E-011-001.

REFERENCES

[1] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, “System architecture directions for networked sensors,” *ASPLOS 2000*, Cambridge, USA, November 2000, pp. 93-104.

[2] Garmin international, “What is GPS”, *http://www.garmin.com/aboutGPS.*

[3] S. Capkun, M. Hamdi, and J.-P. Hubaux, “GPS-free positioning in mobile ad-hoc networks,” *Cluster Computing*, Vol. 5, No. 2, pp 157-167, 2002.

[4] J. Hightower and G. Borriello, “Location Systems for Ubiquitous Computing,” *IEEE Computer*, vol. 34, No. 8, pp. 57-66, August 2001.

[5] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, “The Cricket Location-Support System,” *ACM/IEEE MobiCom 2000*, pp. 32-43, Boston, MA, August 2000.

[6] S. Ray, R. Ungrangsi, F. D. Pellegrini, A. Trachtenberg, and D. Starobinski, “Robust Location Detection in Emergency Sensor Networks,” *INFOCOM 2003*, Vol. 2, pp. 1044-1053, March-April 2003.

[7] D. Niculescu and B. Nath, “DV Based Positioning in Ad Hoc Networks,” *Telecommunication Systems*, Vol. 22, No. 1-4, pp. 267-280, 2003.

[8] N. Bulusu, J. Heidemann, and D. Estrin, “GPS-less Low Cost Outdoor Localization For Very Small Devices,” *IEEE Personal Communications,* Vol. 7, No. 5, pp. 28-34, October 2000.

[9] F. Mondinelli and Z. M. Kovacs-Vajna, “Self Localizing Sensor Network Architectures,” *IEEE Transactions on Instrumentation and Measurement, V*ol. 53, No. 2, pp. 277-283, 2004.

[10] N. Patwari, R. J. O’Dea, and Y. Wang, “Relative Location in Wireless Networks,” *IEEE VTC 2001*, Vol.2, pp. 1149-1153, Rhodes, Greece, May 2001.

[11] P. Bahl and V. N. Padmanabhan, “RADAR: An In-Building RF-based User Location and Tracking System,” *INFOCOM 2000*, Vol. 2, pp. 775-784, March 2000.

[12] J. Hightower, R. Want, and G. Borriello, “SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength,” *UW CSE 2000-02-02*, University of Washington, Seattle, WA, February 2000.

[13] D. Niculescu and B. Nath, “Ad Hoc Positioning System (APS),” *IEEE Globecom 2001*, Vol. 1, pp. 2926-2931, November 2001.

[14] J. G. Lim and S. V. Rao, “A grid-based location estimation scheme using hop counts for multi-hop wireless sensor networks,” *2004 International Workshop on Wireless Ad-Hoc Networks*, pp. 330-334, May 2004.

[15] L. Doherty, K. S. J. Pister, and L. El Ghaoui, “Convex Position Estimation in Wireless Sensor Networks,” *INFOCOM 2001*, vol. 3, pp. 1655-1663, April 2001.

[16] J. P. Sheu, J. M. Li, and C. S. Hsu, “A Distributed Location Estimating Algorithm for Wireless Sensor Networks,” *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing 2006*, Vol. 1, pp. 218-225, June 2006.

[17] L. C. Lin, “A Hop-Oriented Position Estimation Algorithm for Wireless Sensor Networks,” *Master Thesis,* National Taiwan University of Science and Technology, Taiwan, June 2006.