On the Accuracy of Multi-hop Relative Location Estimation in Wireless Sensor Networks

Adel Youssef
Google Inc.
Amphitheatre Parkway
Mountain View, CA 94043
adel@google.com

Mohamed Younis
Dept. Comp. Sc. & Elec. Eng.
University of Maryland
Baltimore, MD 21250
younis@cs.umbc.edu

Moustafa Youssef*
Dept. of Computer Science
University of Maryland
College Park, MD 20742
moustafa@cs.umd.edu

Ashok Agrawala
Dept. of Computer Science
Univ. of Maryland
College Park, MD 20742
agrawala@cs.umd.edu

ABSTRACT

Knowledge of node's locations is an essential requirement for many applications of wireless sensor networks. A major problem with multi-hop location discovery is the accumulated error. In this paper, we analyze the effect of reflection errors and present the Multi-hop Relative Location Estimation (MRLE) algorithm that estimates relative node's positions with low error margins. We capture the impact of different parameters on the accuracy of the estimated position and introduce a new metric, called the CLIQUE factor, that has a very dominant effect on the accuracy of the estimated positions. We further highlight means for trading accuracy for energy consumption and/or computational overhead.

Categories and Subject Descriptors: C.2.2 [Network protocols]: Applications; C.2.3 [Network operation]: Network Management

General Terms: Algorithms, Performance

Keywords: CLIQUE factor, error analysis, multi-hop location discovery

1. INTRODUCTION

In recent years, wireless ad-hoc sensor networks have attracted much interest from the research community as a fundamentally new tool for a wide range of monitoring and data-gathering applications. Sensor networks typically consist of a large number of unattended sensor nodes that are randomly spread over an area of interest. Sensed data are usually sent to a gateway node for processing. Sensor nodes are significantly constrained in the amount of available resources such as energy, storage and communication and computational capacities. Due to the limited radio range and the on-board energy supply, a sensor report is usually forwarded to the gateway over multi-hop paths. Knowing accurate node location, in such a randomly deployed network, is essential in order to correlate the gathered data to the origin of the sensed phenomena. A localization algorithm should determine node's locations in a reliable manner without over-burdening the resource-constrained sensor nodes.

Given the cost and complexity that a GPS receiver may undesirably introduce, significant attention has been paid to *GPS-free* localization [2, 6, 7, 8]. In this case, instead of computing absolute node positions, the algorithm estimates nodes' positions relative to a coordinate system established by a reference group of nodes. Each node typically has the capability to estimate *ranges* (distances) to its neighbors, that are within its communication range. Such proximity estimate is usually subject to some errors. Nodes' proximity estimates are then fused over the entire network to establish a topology with some relative coordinate system.A major problem with multi-hop location estimation is the accumulated error as the node becomes multi-hop away from the gateway node. In [8], we have proposed a two phase scheme, called the local position discovery algorithm (LPD). In the first phase (*initialization phase*), we find an initial position estimate (P_0). In the second phase (*refinement phase*), we use the gradient steepest descent method to refine P_0 and find the optimal relative node positions (P^*) that minimizes the accumulated error.

The focus of this paper is to thoroughly analyze the accuracy of multi-hop relative location estimation and study the different sources of errors; especially *reflection errors*. A reflection error occurs when we have two candidate positions for a node that represent reflection along some axis and we choose the wrong candidate . We show that, during the *initialization phase*, just one reflection error in estimating a node position may lead to a propagation of that error to other nodes. Hence, resulting in an initial position (P_0) with a very high accumulated error. Our experiments show that an error propagation, in the initial position estimate (P_0), cannot be corrected using the computationally-expensive refinement phase. Hence, the best approach to find accurate node position estimates is to avoid reflection error. We present the Multi-hop Relative Location Esti-

*Also affiliated with Alexandria University, Egypt.

IWCMC'07, August 12–16, 2007, Honolulu, Hawaii, USA.

mation (MRLE) algorithm that uses effective heuristics to avoid reflection error. We show that MRLE yields low-error position estimates without a need for the refinement phase. We also introduce a new metric, the CLIQUE factor (CF) that has a very dominant effect on the accuracy of the estimated positions. We further highlight means for using the CF in trading accuracy for energy consumption.

The paper is organized as follows. We start by briefly surveying related work in section II. In section III, we formulate the problem and present the proposed MRLE algorithm. Section IV shows the performance of MRLE via simulations. Finally, section V gives concluding remarks and directions for future work.

2. RELATED WORK

Node localization in sensor networks has been an active research area for the past few years. Published localization algorithms fall into one of three classes or a combination of them. The first class includes *range-free* algorithms, which assume that there is no distance/angle information available at each node [1, 4] . Hence, they use the node's awareness of its neighbors as an indicative proximity measure. In general, *range-free* techniques provide the lowest level of accuracy among the three classes. The second class includes *anchor-based* algorithms [3, 5], where some nodes know their positions usually using GPS. Most of the approaches in this class require a high percentage of anchor nodes in order to reach an acceptable accuracy. Moreover, propagating anchor node location information through the network may lead to a network-wide flooding . Besides, the inclusion of a GPS receiver on each node is not practical.

The third class of localization systems is *anchor-free* (GPS-free) [2, 6, 7, 8]. In this case, instead of computing absolute node positions, the algorithm estimates nodes' positions relative to a coordinate system established by a reference group of nodes. A relative coordinate system can still be transformed to absolute coordinate system by using only three anchor nodes in case of 2-D (or four anchors in case of 3-D). Algorithms in this class can be range-free or range-based. The multi-dimensional scaling (MDS) [7] is an example of *range-free anchor-free* algorithms. Each node computes a local map for nodes that are within 2 hops using mainly node connectivity. Then all the nodes in the network communicate with each other to merge these local maps together to form a global map. The Self-Positioning Algorithm (SPA) [2] is an example of *range-based anchor-free* methods. Each node also builds its own local coordinate system, estimates the positions of *one-hop* neighbors using triangulation and broadcasts this information to all the nodes in the network to build a global network coordinate system.

In [8], we have presented the local position discovery algorithm (LPD), which performs a range-based anchor-free localization. However, instead of forming a local coordinate system at each node like SPA and MDS, LPD builds a network-wide coordinate system only at the gateway node. In this case, the communication overhead to build a global topology is minimal. While other mechanisms consider nodes that are 1 or 2 hops away, LPD estimates the location of nodes that are multi-hop from the gateway. This paper builds on our experience with LPD. We further analyze the sources of localization errors, particularly reflection errors, and present an efficient algorithm that avoid their effect.

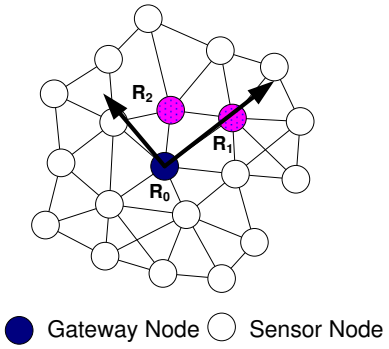


Figure 1: Building relative coordinate system (RCS)

3. MULTI-HOP RELATIVE LOCATION ESTIMATION

We assume that each node estimates distances to its neighbors and reports measurements to the gateway node. The multi-hop relative location estimation problem can be formalized as follows: "Given a multi-hop network where each node knows the distance measurements, perhaps with some high margin of error, between nodes that are within its communication range, the objective is to construct a local map for this network with accurate relative node positions such that the error (E) between estimated and measured distances is minimized."

The error function (E) can be formulated as follows [8]:

$$E(P) = \sum_{i=1}^{n_c-1} \sum_{j=0}^{i-1} (D_P(i,j) - D(i,j))^2 \qquad (1)$$

where n_c is the number of nodes, P is an $(n_c \times 3)$ matrix representing all node positions, and D is an $(n_c \times n_c)$ matrix representing the intra-node distance measurements *as estimated by the nodes*, and $D_p = D(P)$ is a vector function that returns an $(n_c \times n_c)$ matrix representing the *calculated* intra-node distances given a node position estimate P, where for $i, j = 0, 1, \ldots, n_c$

$$D_P[i,j] = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}, \qquad (2)$$

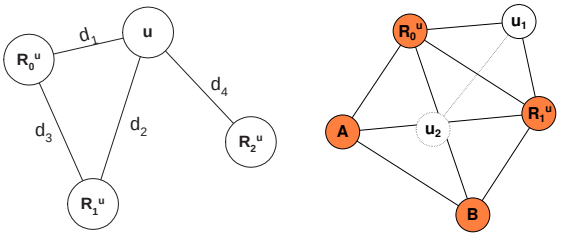
The error function (E) measures the least square error between the estimated inter-node distances (D_P) using P and the actual measured distances measured by nodes (D). The objective is to find the *optimal node positions* P^* that minimizes E.

In order to build a map at the gateway, we need to form a relative coordinate system (RCS). The RCS is defined by fixing three non-colinear nodes (the *reference nodes*): R_0, R_1, and R_2 as shown in Fig. 1. The three reference nodes together form a triangle $\Delta(R_0, R_1, R_2)$. We shall refer to this triangle as the *RCS triangle*. The positions of R_0, and R_1 are given as follows:

$$P_0(R_0) = (0,0), \quad P_0(R_1) = (D(R_0, R_1), 0) \qquad (3)$$

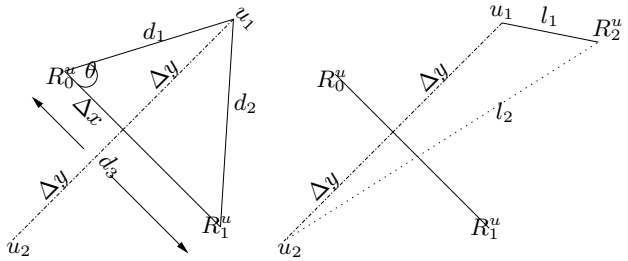
and the position of R_2 can be calculated using the cosine rule.

In [8], different heuristics were proposed to select the RCS. The proposed MRLE algorithm is independent from the heuristic used to select an RCS. However, in the section 4, we shall use the the Minimum Initial Error (MIE) heuristic to select the RCS.



(a) distance to three neighbors known (b) only distance to two non-colinear neighbors known

Figure 2: The two scenarios for estimating the position of a node (u)



(a) Two candidate positions for u (b) selecting one candidate

Figure 3: Estimating the position of a node (u) using three distances

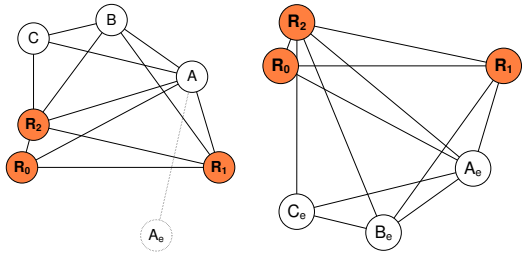
3.1 Multi-hop Relative Position Estimation

Once an RCS is selected, we compute the relative node positions starting from the neighbors of the reference nodes and then moving outward. We shall refer to the set of nodes that have known relative position as the set of *Identified Nodes* (I). It follows that, initially, $I = \{R_0, R_1, R_2\}$. Using the identified nodes, we can calculate the relative position of other nodes that are multi-hop away from the RCS. There are two cases to consider:

1. **A node u knows distances to three** *non-colinear identified neighbors*. Let $(R_0^u, R_1^u, and R_2^u)$ be the identified neighbors of u, as shown in Fig. 2(a), where $R_0^u, R_1^u, R_2^u \in I$. We shall refer to the vector $\overrightarrow{R_0^u R_1^u}$ as *the base line*. Let $d_3 = \| \overrightarrow{R_0^u R_1^u} \|$ be the length of this vector. Let $\overrightarrow{x}$ be the unit vector in the direction of the *base line* $(\overrightarrow{R_0^u R_1^u})$. Let $\overrightarrow{y}$ be the unit vector perpendicular to the *base line* in the direction of R_2^u. Then, using the cosine rule, there are two candidate positions for the node u (u_1 and u_2) where:

$$\cos(\theta) = \frac{(d_1^2+d_3^2-d_2^2)}{(2d_1 d_3)}; \Delta_x = d_1\cos(\theta); \Delta_y = \sqrt{(d_1^2 - \Delta_x^2)}$$
$$\overrightarrow{u_1} = \overrightarrow{R_0^u} + \Delta_x \overrightarrow{x} + \Delta_y \overrightarrow{y} = (u_x, u_y)$$
$$\overrightarrow{u_2} = \overrightarrow{R_0^u} + \Delta_x \overrightarrow{x} - \Delta_y \overrightarrow{y} = (u_x, -u_y)$$

Note that u_2 is the reflection of u_1 across the base line (Fig. 3(a). The third node R_2^u is used to resolve the reflection as shown in Fig. 3(b). Let $d_4 = D(u, R_2^u)$. We choose the candidate that is closer to R_2^u as compared to d_4. The node R_2^u is called the *reflection resolver*. So in the example in Fig. 3(b), u_1 will be chosen since $|d_4 - l_1| < |d_4 - l_2|$. Resolving a reflection is the most important decision in relative position estimation. We shall discuss the issue in detail in the next subsection.



(a) The correct node positions. (b) Effect of initial reflection error in node A

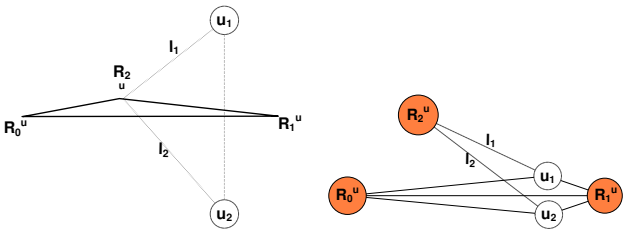
Figure 4: The reflection propagation phenomena.

2. **A node u knows distances only to two non-colinear *identified* neighbors**. Let the two neighbors be R_0^u and R_1^u as shown in Fig. 2(b). Normally this situation happens when the node is on the boundary of the network. Hence, the set of identified nodes (I) contains almost all the internal nodes that are away from the boundary. Using the base line $\overrightarrow{R_0^u R_1^u}$, we can calculate two candidate positions for the node u as discussed before. In order to resolve the reflection and select one candidate, we will use a simple observation. Assume a node v with known relative position and $v \notin \{R_0^u, R_1^u\}$. If the distance between v and candidate $u_i, i = 1, 2$, is less than the node transmission range (T_r) then this contradicts with the fact that v is not a neighbor of u; hence candidate position u_i cannot be the right position. Applying this heuristic to the example shown in Fig. 2(b). The candidate position u_2 is within the transmission range of nodes A and B. Hence, the correct position is u_1.

3.2 The Reflection Error

As discussed earlier, the base line $(\overrightarrow{R_0^u R_1^u})$ is used to calculate two candidate positions for a node u and the *reflection resolver* (R_2^u) is used to select one candidate. If the wrong candidate is selected, the error in the node's position is equal to $2\Delta y$, where Δy is the perpendicular distance between u and the base line as shown in Fig. 3. We shall refer to this error as the *reflection error*. Actually, the reflection error in one node can cause much worse problem, namely the *reflection propagation error*. Consider the subgraph shown in Fig. 4(a). Initially the set of identified nodes (I) contains the three reference nodes. Using the base line $\overrightarrow{R_0 R_1}$, we can calculate the two candidate positions for node A of which one is wrong (A_e). Using the node R_2 to resolve reflection, and noticing that the measured distances contain some error, we can select the wrong candidate (A_e). Now the set of identified nodes contains $\{R_0, R_1, R_2, A_e\}$. We can then estimate the position of node B using $\overrightarrow{R_1 R_2}$ as the base line and node A_e as a reflection resolver. That will lead to estimating a wrong position for node B (B_e) as shown in Fig. 4(b). In a similar way, the reflection error will propagate to node C. What makes the problem worse is that the refinement phase cannot correct this type of error.

The best way to solve this problem is to avoid it! The major cause behind reflection errors is *skinny triangles*. Fig. 5(a) shows an example when the reflection resolver (R_2^u) is close to the base line $(\overrightarrow{R_0^u R_1^u})$. In this case, the difference between l_1 and l_2 is very small; hence; we may not be able to resolve reflection correctly. Another example is shown in Fig. 5(b). Here the node u is close to the base line and again l_1 is very



(a) Node R_2^u is close to the base line (b) Node u is close to the base line

Figure 5: Different cases for skinny triangles

close to l_2. From the two examples, we can notice that a skinny triangle will exist if $\| \overrightarrow{u_2 R_2^u} - \overrightarrow{u_1 R_2^u} \| \leq \delta$ where δ is some threshold depending on the error in the measured distances. In the simulation experiments, we set $\delta = 3\sigma$, where σ is the standard deviation of the range estimation error.

As shown in Fig. 4(a), if the RCS triangle is skinny this may lead to the *reflection propagation error*. Moreover, since the set of *identified nodes* (I) initially contains only those three nodes, the error in the estimated position of all other nodes will be highly affected by the error in the estimated position of the reference nodes (specially R_1 and R_2). Finally, as we move away from the RCS, the accumulated error increases. Based on this it is better to have the RCS near the center of the network. That is why we assume that the gateway node is the origin (R_0) since, it is usually deterministically placed and thus it most probably will close to the center of network. Based on the above observations, the RCS should have the following features:

1. The RCS triangle must not be skinny. We shall use the aspect ratio (AR) of the RCS triangle to measure how skinny it is where the higher the aspect ratio, the skinnier the RCS triangle and vise versa. Assuming the side lengths of the RCS triangle are: d_1, d_2, and d_3. Then the aspect ratio (AR) of the triangle can be calculated as the ratio between the circum radius (CR) and inner radius (IR) of the triangle as follows:

$$s = \frac{d_1+d_2+d_3}{2}; \quad CR = \frac{d_1 * d_2 * d_3}{4 * \sqrt{s(s-d_1)(s-d_2)(s-d_3)}}; \quad IR = \sqrt{\frac{(s-d_1)(s-d_2)(s-d_3)}{s}}; \quad AR = CR/IR; \tag{4}$$

2. The error in estimating the positions of R_1 and R_2 should be minimized. Notic that the position of R_1 is $(D(R_0, R_1), 0)$. Hence, the error in R_1 position depends on the range estimation technology used. For example, if signal strength is used to estimate distances between nodes [4, 5], then as the distance between R_0 and R_1 increases, the error in the distance increases. Hence, it is better to chose R_1 to be closer to R_0, without violating the skinny triangle condition.
3. The RCS should be near the center of the subnetwork. Most of the time, the gateway node will be the origin of the RCS since it is almost the center of the network.

3.3 MRLE Algorithm

In the previous subsection, we have described different features for a good RCS. In [8], we have presented different heuristics to select RCS. MRLE starts by adding R_0, R_1, and R_2 to the set of identified nodes (I), and then iteratively, try to estimate the position of an unidentified node (u) using the technique described in section 3.1. The algorithm terminates when all nodes are identified or we have no more nodes with two or more distances to identified nodes.

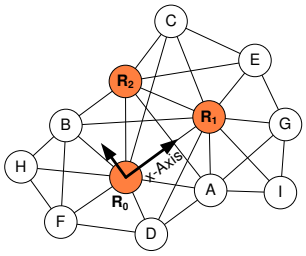


Figure 6: Propagation of node position estimating starting from the reference nodes and moving outward.

In order to limit error accumulation, MRLE assigns to each node an *error level* (EL) indicating how much error the node may have. Assume that the three *reference nodes* (R_0, R_1 and R_2) have an error level of 0 (i.e. $EL(R_0) = EL(R_1) = EL(R_2) = 0$), the error level of a node u ($EL(u)$) can be computed as a function of the error levels of the baseline nodes (R_0^u and R_1^u) as follows:

$$EL(u) = \frac{EL(R_0^u) + EL(R_1^u)}{2} + 1$$

As discussed in section 3.2, a skinny triangle will exist if $\| \overrightarrow{u_2 R_2^u} - \overrightarrow{u_1 R_2^u} \| \leq \delta$. In order to avoid skinny triangles, MRLE selects the base line to be as far as possible from the node u. Now for the selector resolver node (R_u^2), it should be selected as the neighbor node to u with the highest altitude from the base line among all other neighbors. Notice that R_u^2 must also belong to the set of identified nodes. So if there is not a node R_u^2 whose reflection is resolvable, we can postpone estimating the position of node u until more neighbors are identified.

Figure 6 shows an example of MRLE with $I = \{R_0, R_1, R_2\}$. Using the nodes of I, we can calculate the relative position of other nodes that are multi-hop away from the RCS. For example, using the subgraph shown in Fig. 6, since nodes A, B, C know distances to three *non-colinear* neighbors ($R_0, R_1, and R_2$), MRLE can estimate their positions. Hence, the set of identified nodes now include $\{R_0, R_1, R_2, A, B, C\}$. Now we can estimate the position of nodes D and E since they have distances to three or more *identified nodes*. In a similar way, we can calculate the position of nodes F and G, then nodes H and I.

4. EXPERIMENTAL RESULTS

The MRLE algorithm is implemented using MATLAB 6.1. The following parameters are used in the simulation experiments:

1. *Network radius* (k): is the longest path, in terms of the number of hops, between a sensor and the gateway.
2. *Average Node Degree (d):* The node degree is a function of the node transmission range (T_r) and indicates the level of connectivity in the network.
3. *Range error (σ):* is the measurement error associated with each distance between any two nodes. This is dependent on the ranging technology (TOA, AOA, RSSI). In the simulation, we assume that the TOA method is used; hence we assume Gaussian range error with zero mean and variance σ^2.

All experiments were performed over more than 1500 different topologies representing different network sizes (n_c) ranging from 10 to 60 sensor nodes.

The nodes were randomly placed according to a uniform distribution on a 100x100 area. For each topology, the transmission range of each node (T_r) was varied in order to achieve different node connectivity levels (d) ranging from 7 to 17. The network radius (k) ranges from 1 to 5 depending on the network size and node connectivity. The inter-node distance measurements were perturbed with a Gaussian random noise with zero mean and variance σ^2, where σ ranges from 0 to 4. The experiments gear to assess the *location accuracy*, measured in terms of the median error between the estimated positions and the true node positions. Unless noted, in all the experiments below, we use the minimum initial error (MIE) heuristic [8] to select the relative coordinate system (RCS).

The overall goal of the experiments is three folds:

1. We show how does MRLE outperforms the local position discovery (LPD) algorithm in terms of accuracy.
2. Qualify the impact of the network radius and the node degree, as the node becomes k-hops away from the gateway node. The goal is to find parameters that can be tuned to obtain different levels of accuracy.
3. Compare between the accuracy of stand-alone MRLE against MRLE followed by a refinement phase (MRLE+OPT). The goal is to quantify the increase in accuracy when conducting computationally-expensive optimization; hence, develop a policy for trading accuracy for computational power.

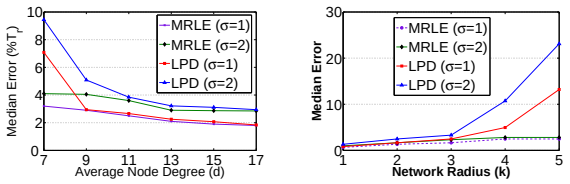
4.1 The Effect of Reflection Errors

In the first set of experiments, we show the effect of reflection error by comparing the MRLE and LPD algorithms in terms of accuracy. Please keep in mind that: (1) LPD does not avoid reflection errors; (2) LPD algorithm uses a computationally-expensive refinement phase to enhance the accuracy of the estimated position while MRLE does not.

Figure 7 shows the location accuracy of MRLE and LPD for different values of node connectivity (Fig. 7(a)) and network radius (Fig. 7(b)). The effect of the range error is also captured in both charts. From Fig. 7(a), it is clear that MRLE outperforms LPD especially for low node connectivity (d) when reflection errors occur with high probability. Also, from Fig. 7(b), we can draw the same conclusion especially when the network radius (k) increases. This is very much expected since the further the node is, the higher the accumulative range error becomes. Also, from both figures, we observe that when using MRLE to avoid reflection errors, the position inaccuracy is bounded while with LPD it increases dramatically for lower connectivity and higher network radius.

4.2 Achievable Accuracy

This set of experiments is dedicated to capturing the effect of network radius (k), node degree and range error on accuracy. Reexamining Fig. 7(a), it can be concluded that increasing node degree has a very positive impact on the overall accuracy but it seems to saturate after a certain level ($d = 13$). Fig. 7(b) shows that an increased network radius worsens the accuracy and magnifies the uncertainty of the results since the accumulated error grows as the node becomes further away from the gateway node.



(a) Changing average node degree (d) (b) Changing network radius (k)

Figure 7: A comparison between MRLE and LPD Algorithm

Combining the observations from figures 7(b) and 7(a), it can be concluded that as the network radius increases, the network connectivity should also be increased in order to maintain high accuracy. Looking into the problem in more details, we can see that the reason for this is that the number of constraints (edges) per node increases; hence; minimizes the possibility of reflection errors. This motivated the need for a new metric that relates the number of edges in the network to the number of nodes (n_c). Note also that the optimal case is when the network is a complete graph, i.e., each node is connected to all other nodes. We thus introduce the *CLIQUE factor (CF)* as a performance metric. The CLIQUE factor measures how close the network is to a complete graph, i.e., it relates the number of edges in the network to the maximum count (optimal case). It turned out that the CF is the major parameter that affects the accuracy. The CF combines the effect of both network radius (k) and node degree (d) regardless of the network size.

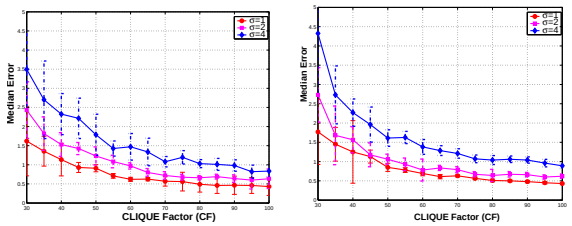
Fig. 8 shows the impact of CLIQUE factor on accuracy for different values of k. The results indicate that the median error is high if the CF is less than 60%. The CLIQUE factor has slight effect on the accuracy beyond 60%. The 95% confidence-interval error bars is almost 0 as the CF increase. For acceptable accuracy, we recommend that the CF be at least 50%. Fig. 8 helps explain the results of Fig. 7. In Fig. 7(a), as the node degree increases, the number of edges increases while the network size (n_c) is fixed. Hence, the CF increases and that is why the accuracy improves as node degree increases. When the network radius increases, Fig. 7(b), the network size increases while fixing the node degree (i.e. number of edges is fixed). Hence the CF decreases and the accuracy deteriorates.

It is also interesting to note that the accuracy can be 60% of range-error standard deviation if the CF is greater than 80%. The accuracy grows as the range error variance increases and can reach up to 75% as CF reaches 100% (i.e., complete graph). This is very important since it offers the design engineer a trade-off between power and accuracy. Figure 9 shows the relation between the CLIQUE factor (CF) and the node transmission range (T_r) for a 100×100 area and a low node density of 0.01.

We like to note that the CLIQUE factor affects the accuracy of the estimated position whether optimization is used or not because increasing the CLIQUE factor reduces the probability of having reflection errors.

4.3 Optimization Factors

In the last set of experiments, we want to show the added accuracy we gain by conducting computationally-expensive least-square error optimization. We compare the accuracy of stand-alone MRLE to MRLE followed by an optional re-



(a) Cluster Radius (k=2) (b) Cluster Radius (k=4)

Figure 8: The effect of CLIQUE factor (CF) on accuracy

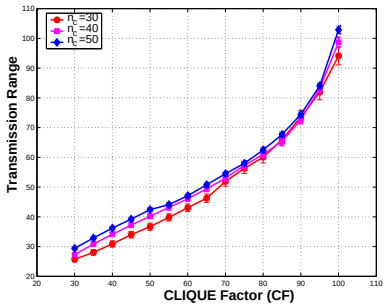


Figure 9: The relation between CLIQUE factor (CF) and node transmission range (T_r)

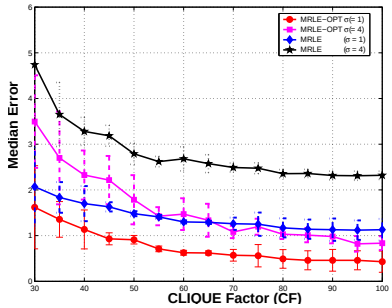


Figure 10: Accuracy before and after optimization

finement phase (MRLE+OPT). Fig. 10 shows how much accuracy is gained by solving the non-linear optimization problem and how it relates to the range-error standard deviation (σ). It is clear from the figure that the refinement step increases the accuracy more than 100%. This gives the design engineer a parameter to tune, namely the node computation overhead, since the non-linear optimization during the refinement step involves significant computation. An interesting observation to make about Fig. 10 is that the error in the estimated position using MRLE is almost the same as the range error. The enhancement is much better as range error increases and the error in the estimated position can reach approximately 50% of the range error. This confirms the effectiveness of the MRLE algorithm and show how accurate the initial estimated position is.

Finally, Table 1 lists different accuracy levels for range error $\sigma = 4$. Each row represents different CLIQUE factor (i.e. different transmission power) while the two columns indicate the computational power based on whether optimization is performed or not. The table hints on the possible trade-off between computational power and accuracy that can be exploited by the application.

CF	Low (MRLE)	High (MRLE+OPT)
Low (40%)	.25	.25
Medium (60%)	.75	.5
High (100%)	.25	.25

Table 1: Trading accuracy with computation capacity and transmission power

5. CONCLUSIONS AND FUTURE WORK

A major problem with location discovery in wireless sensor networks is the error accumulated in the node position as it becomes multi-hop away from the gateway node. In this paper, we have identified reflection errors as the major cause of accurate position estimation . We have presented MRLE, a GPS-free range-based localization algorithm that estimate nodes' positions with low error margins. The performance of MRLE has been validated through simulation. Our experiments have captured the impact of the different parameters, such as network connectivity, on the accuracy of the estimated position. We have introduced a new metric, called the CLIQUE factor, which has a very dominant effect on the estimation accuracy regardless of the network size.

We have further pointed out a tradeoff between localization accuracy and both transmission power and computational overhead. The application designer can exploit such tradeoff in order to tune the network operation to desired performance. We are currently extending the approach to clustered network architectures, in which multiple gateways are present. We are investigating the potential collaboration among the gateways and the formation of a global coordinate system using a number of cluster-based ones.

6. REFERENCES

[1] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less Low-cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications*, 7(5):28–34, October 2000.

[2] S. Capkun, M. Hamdi, and J.-P. Hubaux. Gps-free positioning in mobile adhoc networks. In *Hawaii International Conference on System Sciences (HICSS-34)*, pages 3481–3490, January 2001.

[3] X. Cheng, A. Thaeler, G. Xue, and D. Chen. Tps: A time-based positioning scheme for outdoor sensor networks. In *the Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, March 2004.

[4] D. Niculescu and B. Nath. Ad-hoc positioning system. In *the Proceedings of IEEE Global Communication Conference (Globcom'01)*, November 2001.

[5] A. Savvides, C. C. Han, and M. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *the Proceedings of the ACM Conference on Mobile Computing and Networks (MOBICOM'01)*, July 2001.

[6] Y. Shang and W. Ruml. Improved mds-based localization. In *the Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, March 2004.

[7] Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P. J. Fromherz. Localization From Mere Connectivity. In *the Proceedings of ACM MOBIHOC 2003*, pages 201–212, Annapolis, MD, June 2003.

[8] A. Youssef, A. Agrawala, and M. Younis. Accurate Anchor-Free Localization in Wireless Sensor Networks. In *the Proceedings of the 1st IEEE Workshop on Information Assurance in Wireless Sensor Networks (WSNIA 2005)*, Phoenix, Arizona, April 2005.