Positioning Algorithms
Semester Thesis

Christina Pöpper

August 1, 2003

ETH Zurich
Department of Computer Science
Distributed Computing Group

Prof. Dr. Roger Wattenhofer

Tutors: Ruedi Arnold
Regina Bischoff

Abstract

The present semester thesis deals with the problem of position determination in a mobile ad-hoc network without a common server infrastructure. The question is how exact a node which does not know its position in such a network can compute it by receiving information from other nodes. A certain subset of nodes, called the anchor nodes, are assumed to know their positions. The position determination has been analysed for the one-dimensional and the two-dimensional case. The objectives were to find theoretical limits for these two cases under certain conditions. For one dimension an explicit approximation with bounded error is given. For two dimensions a simple algorithm as well as variations and enhancements of it are discussed.

Contents

1 Introduction

Determining the position of nodes in a mobile ad-hoc network is interesting for a number of reasons, it may for example be helpful to know the position of a person equipped with a mobile device who has had an accident and is unable to communicate at a given moment.

Devices, called nodes from now on, can send and receive signals only within a certain distance, the transmission radius r.

Assuming that determining the distance between two nodes only by signal reception will always be error-prone, there are (at least) two models possible, see Figure 1. In the first one, each node can be in a binary state to every other node: either it receives a signal from the other node, which implies that it is situated within the transmission radius, or it does not receive any signal. In the second model, a node can determine its distance from another node with a certain error δ, e.g. by evaluation of the signal strength. It is assumed that no other criterion, such as angle determination of the signal, can be used.

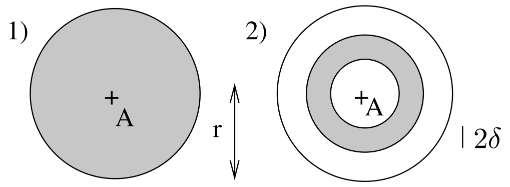

Figure 1: Two models: In 1) the node can be situated anywhere within the transmission radius, in 2) it lies within a ring around anchor node A.

The analysis in this paper will be restricted to the first case. The reason for this is that in the one-dimensional case the error of the computed position can be reduced by the second model, but the algorithms will not be simpler, because the same steps have to be taken. In the two-dimensional case the distance containing an error complicates the computation and does not help much to reduce the position error, because the path of hops from node to node is not straight in general and after a few hops only the maximal distance is known (see Figure 2).

Throughout this paper the *position interval*, continuous in space or not, denotes the area where a node may be situated due to the received information. When referring to the *maximal error* it is assumed that the centre of this interval is chosen as the most likely position of the node. The centre itself may lie outside the interval. Both the position interval and the maximal error are means to express the accuracy of position determination.

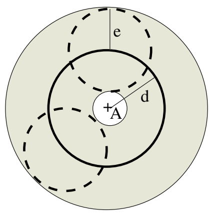

Figure 2: Second Model: Node q has distance $d \pm \delta$ from A. Node p has distance $e \pm \delta$ from p. But the area where p may lie (grey zone) is not bounded in δ.

2 Hop Algorithms

In a network of nodes, the trip of a message from one node directly to another node is denoted with the term *hop*. The *minimal number of hops* between two nodes p and q equals the minimal number of messages between different nodes such that information from p to q can be transmitted. The number of nodes on this minimal path equals the minimal number of hops plus one.

For both one and two dimensions, the algorithms presented in this paper construct a position interval from the hop information. This information mainly contains the number of hops from anchor nodes, whose positions are known (in advance or sent in the messages). See Figure 3 for an exemplary interval.

We call these algorithms *hop algorithms.* Nodes receive the number of hops from other nodes by nodes within their transmission radius and send this information to their neighbours, which is all information a node can use for its position determination. Uncomplicated hop algorithms may not be optimal, but the messages between nodes are small and the computation may be quite simple.

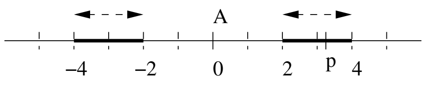

Figure 3: Illustration of a position interval in one dimension: Interval (indicated by the arrows) where node p, lying four hops away from A, may be situated. The transmission radius is assumed to be 1.

2.1 Simple Hop Algorithm

The process of position determination consists of roughly two phases:

1. Information spreading
2. Local computation

In the first phase the necessary information is spread in the network, reaching node p which wants to determine its position. p itself takes part in the dissemination of information. The information might be updated later on and the position may be recalculated.

The first phase for a simple hop algorithm can be as follows (as p sees it):

> Whenever p receives a new message it stores the contained information and sends the message out to all nodes within the transmission radius, increasing the number of hops by one.

In the second phase p computes its position locally by the information it received in the first phase. The simple hop algorithm determines p's position interval. The centre of this interval is then returned as p's position, minimizing the maximal error. In the following subsection we discuss more details about errors.

2.2 Comparison of Algorithms

In order to draw a comparison between the simple hop algorithm and other algorithms it is necessary to define the quality of a positioning algorithm. There are two approaches of when an algorithm A_1 is better than an algorithm A_2.

1. **Interval.** The position interval determined by A_1 is smaller and therefore the maximal possible error is less than or equal to the determination of A_2.
2. **Error.** The algorithms return a specific position. The position returned by A_1 is on average nearer to the real position than the return value of A_2.

These two definitions are *not* equivalent.

For the comparison by **interval** size a better algorithm is always at least as good as the simple hop algorithm, because additional information can never enlarge the position interval. The consideration of interval sizes makes simple comparing possible.

The comparison by **error** is more complicated, because an algorithm being better averaged over all settings does not involve a smaller error in a specific setting (see Figure 22 in Section 4.4 for an illustration). The returned position depends on the specification of the second phase of the algorithm, such as (the first two require a position interval determination):

- centre of the specified position interval (compare the simple hop algorithm)
- random value within the position interval
- position of the nearest anchor node
- centre of the convex hull of the nearest n anchor nodes.

As a general remark on errors it is worth mentioning that the more nodes there are in the setting, the less the error of the position determination will be potentially, because then the minimal number of hops from an anchor node to the node in question will be small and the maximal distance is more restricted. The path becomes straighter. This applies to the one-dimensional case, but is even more important for the two-dimensional case as we will see in Section 4. Besides, the more nodes there are in the setting, the fewer nodes receive no or only one message from an anchor node. Receiving no or only one message is not only a problem at the boundaries of the setting, but also within when there are no more nodes in the transmission radius.

3 The One-Dimensional Case

In this section we consider the simple hop algorithm in one dimension. Maximal interval sizes and errors will be derived depending on the distance between anchor nodes and on the number of hops.

3.1 Accuracy of Position Determination

Assume that node p wants to determine its position. There are several possibilities for the number of nodes within its transmission radius (number of neighbours):

1. p receives no message.

   Position determination (better than guessing) is not possible. So the simple hop algorithm is as good as the best algorithm.

2. p receives a message from one anchor node A.

   According to the simple hop algorithm p gets the minimal number n of hops from A, i.e. the number of hops of the path with the fewest nodes. The maximal distance of p from A equals nr, when all hops cover the total transmission radius r. The minimal distance is bounded since two nodes being no direct successors on the path from A to p must have a distance of at least r. Otherwise the minimal number of hops would be less. So the average distance between two nodes on the path is at least $r/2$. Consequently, the minimal distance between p and A equals $\lfloor nr/2 \rfloor$. Then the maximal inaccuracy of the hop algorithm is two intervals, each of size $\lceil nr/2 \rceil \approx nr/2$ (see Figure 3). Hence the returned position will be the position of the anchor node, yielding a maximal error of nr.

3. p receives messages from at least two anchor nodes.

   p gets the minimal number of hops from the anchor nodes. Assume that p considers the two smallest numbers of hops, m and n. Let d be the distance between the anchor nodes A and B. See Figure 4.

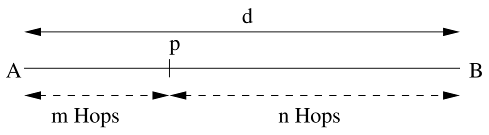

Figure 4: General setting for a node p between two anchor nodes A and B at distance d.

- If the determined intervals for p from A as well as B overlap only between the two anchor nodes, p's position interval can be reduced to at least $d/3$ (worst case). Choosing the position of p in the middle of this interval, the maximal error equals $d/6$. For a proof see Section 3.2.
- If p cannot determine whether it lies between A and B, there are two cases possible (see Figure 5). Let $m < n$.

  In the first case (for $mr < d \Rightarrow p$ may be located between A and B) two intervals for p's position exist. By guessing that p lies in the middle of these intervals, the maximal error equals d/3. For the maximal error, both intervals are of maximal size d/6. For a proof see Section 3.3.

  The second case (for $mr > d \Rightarrow p$ cannot be located between A and B) can, as an approximation, be considered as the situation when p receives a message from only one anchor node. A and B are close to each other compared to the distance to p. The positioning error is not bounded by d.

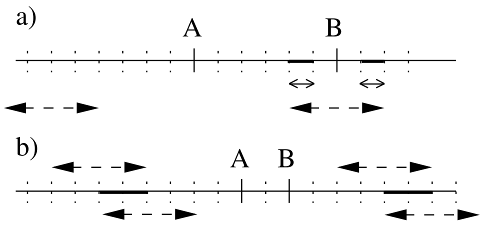

Figure 5: In a) p may be situated between A and B (thick lines). In b) one may consider A and B conflated to one anchor node because p is located far away compared to the distance of A and B.

Summarizing, the following table for p's position can be derived (worst case analysis):

# received messages	interval size	maximal error
0	unbounded	unbounded
1	2 intervals $nr/2$ each	nr
2, p between	$d/3$	$d/6$
2, p not ", case a	$2 \times d/6$	$d/3$
2, p not ", case b	cmp. 1 mess.	$< nr$,cmp.1 mess.

3.2 Derivation of p's Interval between two Anchor Nodes

p lying between A and B can be determined by considering additional information from other nodes (in rare cases) or by the specific overlapping of the areas of both A and B:
$max(m,n) \cdot r < d + min(m,n) \cdot r/2$.

There are four cases how the position interval I can be bounded by the minimal (top line) and maximal (bottom line) distance of p from A and B (see Figure 6):

$$I = d - \left\{ \begin{array}{c} m \cdot r/2 \\ d - n \cdot r \end{array} \right\} - \left\{ \begin{array}{c} n \cdot r/2 \\ d - m \cdot r \end{array} \right\}$$

Simplifying and adding the conditions yields for I

$$\left\{ \begin{array}{ll} d - (m+n)r/2; & d/r \leq min(m/2+n, m+n/2) \\ n \cdot r/2; & m/2+n < d/r < m+n/2 \\ m \cdot r/2; & m+n/2 < d/r < m/2+n \\ (m+n)r - d; & d/r \geq max(m/2+n, m+n/2) \end{array} \right.$$

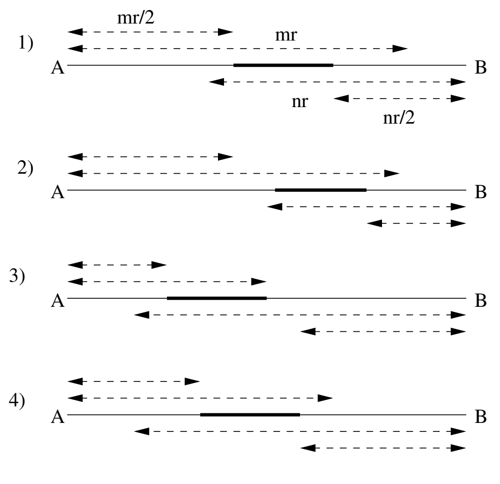

Figure 6: The interval (bold line) can be bound by four different combinations of hop-distances.

When will the interval I be maximal? The overlapping of the intervals from A and B is maximal when the interval determined by A is not reduced by the interval determined by B and vice versa. This happens only for $m = n$. The equation for I is simplified (case 2 and 3 are impossible) to:

$$I = \left\{ \begin{array}{lll} d - m \cdot r; & for & d/r \leq 3/2m \\ 2m \cdot r - d; & for & d/r \geq 3/2m \end{array} \right.$$

Solving for I yields $I \leq d/3$ for both cases. This situation is presented in Figure 7.

Thus, the main conclusion of this subsection is:

Theorem 1 *If node p lies between two anchor nodes, its position can be computed with the maximal error $d/6$, where d is the distance between the anchor nodes. The maximal size of the interval is $d/3$.*

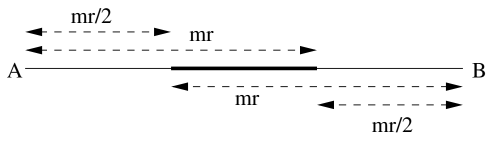

Figure 7: Maximal interval if $m = n$: this leads to one third of the distance between A and B.

3.3 Derivation of p's Interval not surely between two Anchor Nodes

Consider the case where p may be situated between or outside of two anchor nodes. See Figure 5a.

The two intervals are maximal if the intervals given by A and m are not reduced by the number of hops n from B (with $m < n$). This yields the following two conditions:

$$nr/2 \leq d - mr \qquad (1)$$
$$nr \geq d + mr \qquad (2)$$

Solving for n yields $d/r + m \leq n \leq 2(d/r - m)$.

This bounds m as well: $m \leq \frac{1}{3}d/r$.

As mentioned, the intervals are limited by the number m of hops from A. Consequently, the intervals are of size $mr/2 \leq d/6$ each. The maximal error is minimized for choosing p's position in the middle of these two intervals, at A's position. This yields a maximal error of $mr \leq d/3$.

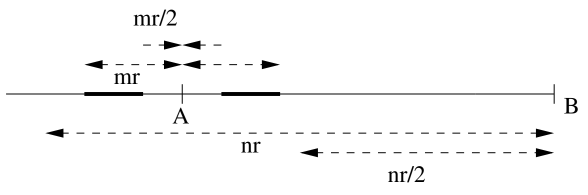

Figure 8: The two intervals of A are not reduced by B when both the minimal and the maximal distance from B are outside of the two intervals.

3.4 Further Analysis

Up to now the hop information from at most two anchor nodes was included in the computation and considered in the analysis. In most cases this gives the best possible result, because p lying between A and B implies that the path from a more distant anchor node does not yield any additional information. But if it is not sure that p is positioned between the two anchor nodes, the information from a third anchor node may be helpful (if p receives a message from one more). See Figure 9 for an example.

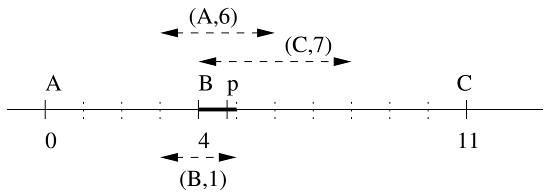

Figure 9: The hop information from a third anchor node (C), which is further away than the first two anchor nodes (A and B), can yield additional information and reduce p's interval. (A,6) signifies that the minimal number of hops from A to p is 6.

The simple hop algorithm can be altered to improve its accuracy.

One alteration affects mainly the case where p receives only a message from one anchor node. In this case the position can be computed more precisely if p knows on which side of the anchor node A it is located. In general there is no possibility for a hop algorithm to detect this but there are cases that allow this determination: If node q lies within the distance of one hop from A, and if q was able to determine on which side of A it is located by receiving information from another anchor node, then A can send this information to all nodes within the transmission radius. If a node directly receiving this information does not hear q then it knows that it is located on the other side of A. This information then needs to be propagated enabling p to reduce its position interval, but enlarging the messages and needing a second pass.

4 The Two-Dimensional Case

4.1 Differences between One and Two Dimensions

The observations obtained from the one dimensional case cannot simply be taken over for two dimensions.

In the one-dimensional case the position intervals are segments of a straight line, whereas in the two-dimensional case they are circles for a message of one anchor node and take on different two-dimensional shapes for the overlapping of intervals for several anchor nodes.

The main difference between the dimensions can be stated as follows: In one dimension the minimal number of hops between a node p and an anchor node A implies a certain minimal distance, namely $nr/2$. Consequently, in one dimension, a node p is farther away from A than all nodes on the path from A to p.

This is not the case in two dimensions (see Figure 10): The number of hops contains only information on the maximal distance, which is nr analogous to one dimension. But the minimal distance is $r + \epsilon$ ($\epsilon > 0$) if the minimal number of hops exceeds one and p does not receive the message of the anchor node directly in one hop.

This affects the knowledge of the surroundings of a node: In one dimension, if a node receives a message from an anchor node A, then it certainly receives messages from all anchor nodes between A and p. This does not hold for two dimensions (see Figure 11).

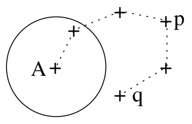

Figure 10: In two dimensions, the node p can be farther away from A than q although p's number of hops is smaller (3 compared to 5 for q).

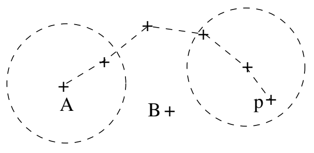

Figure 11: p receives a message from anchor node A. There can be nodes between A and p, such as B, for which there does not exist a path to p.

4.2 Observations and Results

The first two observations concern the simple hop algorithm:

Shape of position intervals. As mentioned, the position interval due to one anchor node is a circle. Thus, the position interval is the overlapping of circles of all anchor nodes reached by a node p and in which p could be positioned. As circles are convex and intersections of convex forms conserve this property, all position intervals are *convex* as well as *continuous.* As the only exception there may be circular holes in this area, because a node is at least at a distance r from an anchor node if it does not receive the message directly in one hop.

Maximally stretched paths. *Maximally stretched paths* denote paths on which all hops are of length r and all hops are on a straight line. The feature of those paths is not detectable by the hop information from one anchor node generally. But when a node p lies on the connection line of two anchor nodes A and B and there exist maximally stretched paths from both A and B to p, then p's position is determined exactly. This plays an important role for the hop algorithm, even if the paths are only roughly maximal stretched, and will be used in the next sections.

The following observations apply to all hop algorithms, not restricted to the simple hop algorithm:

Intersections. Let us assume that a certain algorithm provides node p with the information that there is a path between X and Y.

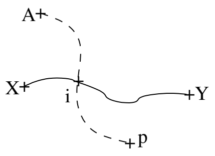

Figure 12: Intersection of two paths.

The question is whether the algorithm can determine if the path from an anchor node A to p traverses the path between X and Y, i. e. whether there is an intersection, meaning that A lies on the opposite side of the X-Y-path compared to p. This could be of importance for restricting p's position interval.

An intersection can exist when two paths either share a common node or when at least one node from a path hears at least one node from the other path.

Two statements concerning intersections are as follows:

Firstly, no hop algorithm can detect definitely all intersections. In Figure 13 an example is presented to prove this statement.

Secondly, a hop algorithm can determine that there is *no* intersection, namely when there is no node on a path that has a neighbour being part of the second path. This situation cannot be revealed by the simple hop algorithm, but a more refined algorithm may detect this if the two paths are connected by other nodes. This can be used for the position determination of a node. In those cases, the more refined algorithm can certainly reduce the position interval compared to the simple hop algorithm. Nevertheless, it is not sure, that the returned position is nearer to p's real position.

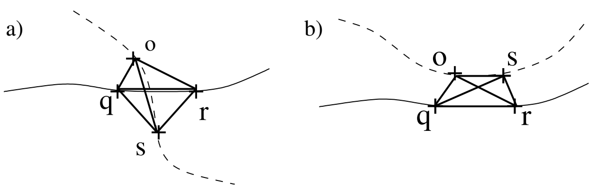

Figure 13: For all hop algorithms, these two situations with intersection (a) and without (b) are indistinguishable, because all nodes receive messages from the same neighbours (indicated by the bold straight lines).

Number of anchor nodes providing information. The question in this subsection is whether there exists a (finite) number of anchor nodes with a specific arrangement such that no new anchor node N outside of the existing anchor nodes can determine the position of node p inside more precisely.

Then, the following statement is valid for all hop algorithms:

Theorem 2 *Taking a new anchor node N into consideration may reduce the position interval of a node p, regardless of how many anchor nodes have already been considered.*

Proof. Assume there exists an upper limit u for the number of anchor nodes, such that a new anchor node outside the convex hull of these existing anchor nodes can provide *no* additional information for the determination of p's position. We will construct a counterexample.

In this counterexample, p's position cannot already be determined exactly by the existing anchor nodes; otherwise the consideration of a new anchor node is useless anyway. Moreover there exist (roughly) maximal stretched paths from an existing anchor node to p as well as from N, the new anchor node, to p. Besides, *no* existing anchor node is part of the path from N to p; otherwise N involves no new information.

It can not be ruled out that the existing maximally stretched path and the path from N to p are on a straight line, which locates p precisely (or more precisely if there are only *roughly* maximal stretched paths). This is valid independent of the limit u. It is therefore worth considering the new anchor node, regardless of how many anchor nodes have been taken into account already. See Figure 14 for an illustration. $\square$

This implies that a good hop algorithm must in general not neglect anchor nodes from the start if it has already processed the information from a certain number of anchor nodes.

4.3 Simple Hop Algorithm and Enhancements

The simple hop algorithm for two dimensions is defined as in Section 2.1.

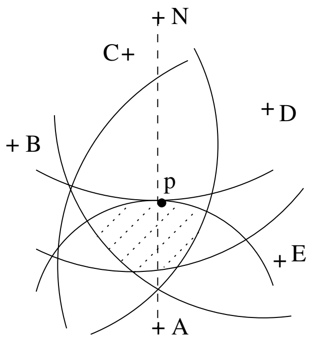

Figure 14: The anchor nodes A, B, C, D, E locate p up to the dotted area. There is a maximally stretched path (dashed line) from A to p. The new anchor node N has a maximally stretched path to p as well and thus determines p's position exactly together with the information from A, regardless of the number of anchor nodes.

As the position intervals in two dimensions are in general considerable in size the aim is to alter the hop algorithm in order to decrease the position intervals while increasing the number and size of messages only as little as possible.

Besides the number of hops from an anchor node, what else can be used to determine the position of a node?[1]

Two possible alterations are considered in the following: the first upgrades the simple hop algorithm by running it several times so that nodes which were able to locate their positions exactly or only with a tiny error are treated as additional anchor nodes. The second alteration takes merge information into account. This alteration may improve the position determination in the order of the square root of the hop distance from an anchor node.

4.3.1 Iteration

In the course of the simple hop algorithm some nodes may be able to determine their positions precisely. It makes sense to treat them as anchor nodes in order to help other nodes determine their positions. The more anchor nodes there are, the more accurately the positions can be determined potentially. The expense of this is that the simple hop algorithm has to be executed several times, where the

[1]Except the signal strength and angles, which are not considered here.

nodes which determined their positions precisely or with a tiny error now act as anchor nodes. Iterating the simple hop algorithm has no effect on the messages sizes.

4.3.2 Merge-Information

For positioning node p let us assume that there are at least two paths from two different anchor nodes A and B which merge at some node M on their way from the anchor node to p. Let both of these paths be of length n. M is called the merge node.

Merge information denotes additional hop information, namely the number of hops from both A and B to M.

Symmetry (number of hops from A to M = number of hops from B to $M = m$) is assumed here for simplicity. M is a node on the path, consequently $1 \leq m \leq n-1$.

Then the following can be stated:

Theorem 3 *By using merge information, the maximal error of p's position, determined by the simple hop algorithm, can be improved in the order of $O(\sqrt{n})$, where n is the number of hops from both A and B to p.*

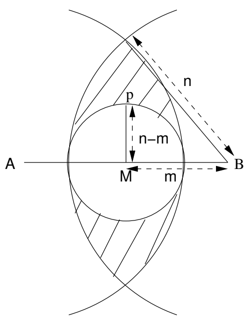

Figure 15: A and B are two anchor nodes. The blank circle represents the area where p may be situated computed by an algorithm which uses the merge information. The striped area represents the additional area under the computation of the simple hop algorithm (n = number of hops from either A or B to p).

Proof. A setting will be constructed in which the error of p's position is improved in the mentioned order. See Figure 15.

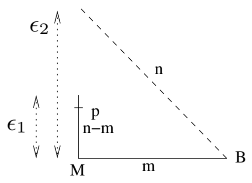

Figure 16: When M is determined exactly, the merge information makes the maximal error be $\epsilon_1 = (n-m) \cdot r$. Without merge information (dashed line) the maximal error is ϵ_2 when p's position is determined in the middle of the interval limited by n hops from A and B (middle of the two circular arcs in Figure 15).

Consider the setting in which A and B determine the position of merge node M exactly.[2] This yields the maximal improvement by merge information.

We are interested in the maximal error ϵ made when determining p's position. See Figure 16.

By using merge information, ϵ equals $\epsilon_1 = (n-m) \cdot r$. The simple hop algorithm without merge information has an ϵ equal to $\epsilon_2 = \sqrt{n^2 - m^2} \cdot r$.

The ratio R of the error without merge information to the error with merge information is

$$R = \frac{\epsilon_2}{\epsilon_1} = \sqrt{\frac{n+m}{n-m}}$$

This ratio is maximal for $m = n-1$, then $R = \sqrt{2n-1}$.

This means that the maximal error can be reduced by using merge information. The maximal reduction is in the order of $\sqrt{n}$. □

The simple hop algorithm uses only the shortest paths from anchor nodes for the position determination. In contrast to this simple approach longer paths with merge nodes may yield information (see Figure 17 as an example):

Longer paths with a merge node may reduce the interval that was determined by the hop information of the shortest paths.

[2]In this extreme case, where M's position is determined exactly, the same improvement can be achieved by iteration.

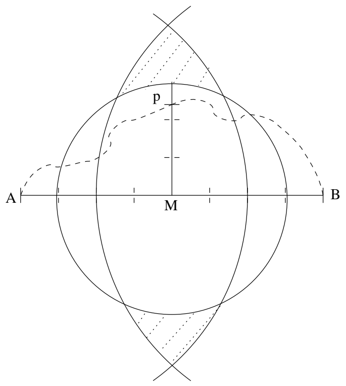

Figure 17: The minimal paths from either A or B to p (dashed lines) yield the two circular arcs as margins for the position interval. By longer paths (straight lines), which determine the position of a merge node M exactly, the interval for p is reduced by the dotted area.

The message size increases, because M must include the information on its position on the path. The size of the messages is increased only by a constant factor, because maximally one merge information per hop information from an anchor node has to be added, specifying how many hops there are between the anchor node and M. If the accuracy of M's position decreases, so does the accuracy of p's position.

4.3.3 Selected Hop Algorithms

As mentioned in Section 2.2 there are different specifications of a hop algorithm for returning the position of a node. Some are discussed below.

Nearest-Anchor The simplest algorithm chooses the position of the anchor node for which the number of hops m is minimal. Its running time is short, but the maximal error mr is unbounded compared to better algorithms, because a better algorithm may determine the position exactly in specific settings.

k-Nearest-Anchors Generalising, anchor nodes up to an arbitrary number k are considered. There are two criteria which k anchor nodes should be taken:

1. Shortest numbers of hops.
2. Combinations of two anchor nodes, such that both paths from the anchor nodes to p are as stretched and straight as possible.

The second criterion is detectable by comparing the ratios of the distance between pairs of anchor nodes to the maximal length of both paths from the anchor nodes to p.

After choosing the anchor nodes, the algorithm determines the position interval of node p and chooses the centre of this interval as the position. In many cases the i-Nearest-Anchors algorithm returns a more accurate position than the j-Nearest-Anchors algorithm for i>j, but there are settings where the simpler algorithm returns the better position (see Figure 18). It depends on the setting and if the minimal number of hops actually indicates the shortest distance. The greater k, the more computation is necessary for the weighted position.

The error may even be higher than mr: up to $2mr$. See Figure 19.

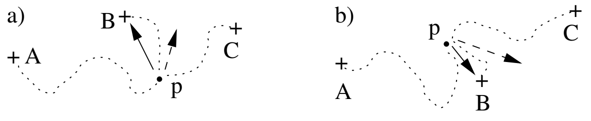

Figure 18: p is situated at the dot, the arrows indicate the positions where the different algorithms assume p to be (straight arrow → Nearest-Anchor algorithm; dashed arrow 2-Nearest-Anchor algorithm. In a) the 2-Nearest-Anchors yields the better result, in b) the Nearest-Anchor is better.

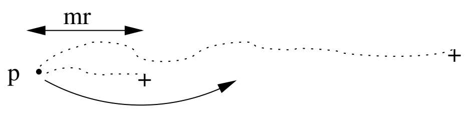

Figure 19: The 2-Nearest-Anchors algorithm determines a position which may have an error higher than mr.

4.4 More Precise Algorithm

4.4.1 Specification

The observations of the previous subsections (no intersections, iteration and merge information) lead to defining an algorithm that makes use of these discoveries and thus can determine a node's position interval more precisely.

The algorithm needs to know all paths for being able to rule out intersections. Thus p must receive the neighbour information (all nodes that are situated within the transmission radius of a node) for all nodes there exists a path to. This is some kind of global information. This new algorithm can exploit the fact, that nodes receive *no* messages from each other.

The setting for this algorithm can be considered as a graph. The question is how exact the position of a node can be determined if it is known which nodes are connected by an edge of length less than or equal to r in the graph, given that the anchor nodes have a fix position.

In more details, the algorithm can look like (for an illustration see Figure 20):

1. All anchor nodes send their ID and position.
2. Repeat: All nodes receiving a message update their internal storage of the state of the graph and send this information to all neighbours.
3. The nodes compute their positions locally with the global information which nodes are neighbours.

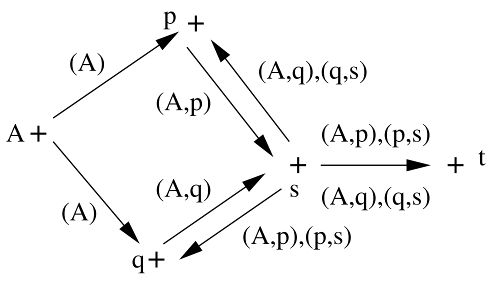

Figure 20: Intermediate state of the algorithm. t has already got the global information which nodes are neighbours and can e.g. compute the number of hops. (A,p) means that the distance from A to p is at most r.

4.4.2 Comparison to the Simple Hop Algorithm

The above specified algorithm is always better than or at least as good as the simple hop algorithm when the interval size is used for comparison. But this does not imply that it returns a more accurate position in every setting, as was already mentioned in Section 2.2. For two counterexamples see Figures 21 and 22.

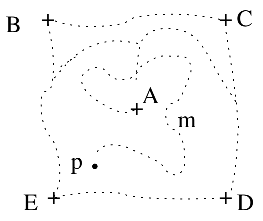

Figure 21: Assume that there are almost maximally stretched paths between the outer anchor nodes forming a "border". The algorithm of this section determines that the path from A does not cross the border. Thus, the algorithm determines that p is surely situated within the quadrilateral. The simple hop algorithm knows that p receives messages from all anchor nodes. The centre of intersections of all five position intervals from A, B, C, D, and E lies within the quadrilateral as well. Hence, the simple hop algorithm potentially is *not* worse than the refined algorithm.

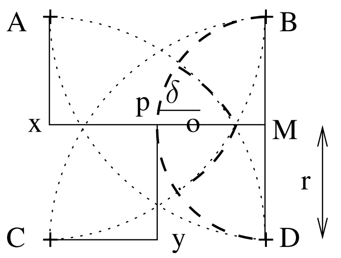

Figure 22: The position interval for p determined by the simple hop algorithm with A, B, C, and D as anchor nodes is the dotted innermost closed shape around p. The returned position is its centre. As p, in this setting, is actually situated in the middle, the error is 0. By the refined algorithm p knows not only the number of hops from the anchor nodes, but it can also determine M's position exactly because of the maximal distance r from B and D to M. Hence, p can reduce its position interval. To make the maximal error minimal p's position is determined where "o" can be found, making an error $\delta > 0$. In this case, the new algorithm, which is better than the simple hop algorithm averaged over all settings, is worse in regards to the made position error. Nevertheless, its maximal error as well as its interval size are still less.

5 Conclusion

In one dimension the setting of anchor nodes can be chosen in such a way that, given a dense distribution of nodes, a node situated between two anchor nodes can compute its position with an error of $d/6$, where d is the distance between the anchor nodes. Hence, given a maximal error δ, the anchor nodes should be established at regular distances of $d = 6 \cdot \delta$. This is no guarantee for the maximal error, because a node may receive no message at all, but the more nodes there are in the setting the higher is the probability of receiving a message.

For the two-dimensional case there is in general no limit for the maximal error or the interval size as a function of the distance between two or more anchor nodes. Although we did not find an algorithm which returns a significantly better position than the simple hop algorithm or minor variations of it in *all* settings, we specified several approaches for a better algorithm.