Distributed Mass-Spring-Relaxation for Anchor-free Self-localization in Sensor and Actor Networks

Juergen Eckert*, Felix Villanueva†, Reinhard German*, Falko Dressler‡
*Computer Networks and Communication Systems, Dept. of Computer Science, University of Erlangen, Germany
†Computer Architecture and Networks, School of Computer Science, University of Castilla-La Mancha, Spain
‡Institute of Computer Science, University of Innsbruck, Austria
{juergen.eckert,german}@informatik.uni-erlangen.de, felix.villanueva@uclm.es, dressler@ieee.org

***Abstract*—We present a fully self-organizing approach for creating and maintaining a reference coordinate system for self-localization in Sensor and Actor Networks (SANETs). GPS technology has become a *de facto* standard for outdoor localization, however, self-localization in GPS denied scenarios is still extremely challenging. Typically, anchor nodes or global state information are used to update the nodes' location information. In contrast, we present a fully self-organizing strategy to generate a distributed reference coordinate system. In particular, we use autonomous robot systems to span and maintain this coordinate system. In particular, we investigated the capabilities of the Mass-Spring-Relaxation (MSR) algorithm, which is frequently used for fault-tolerant and robust localization. Unfortunately, this algorithm needs certain globally valid state information. We extended the MSR algorithm in two ways: First, we made the algorithm independent of *a priori* global knowledge, and, secondly, we provide extensions that make the algorithm more reliable and robust, and to reduce the number of necessary information exchanges between the nodes. As can be seen from the simulation results, our advanced MSR is very accurate and clearly outperforms the classical MSR for increasing network sizes. We also validated the simulations in an experimental setting. The obtained results confirm the very high localization accuracy.**

I. Introduction

Localization is still one of the most challenging aspects in the domain of Wireless Sensor Networks (WSNs). Topological information is the basis for many routing and data management algorithms; and accurate geographic positions are strongly required for a large set of additional services including autonomous robot navigation, geographic routing, geo-correlated information systems, as well as to support software distribution and target tracking [1], [2]. In the scope of this paper, we focus on an Autonomous Localization Framework (ALF) (see Figure 1) consisting of a self-organizing set of mobile robots forming a coordinate system (step 1 in Figure 1) in order to autonomously explore and expand into unknown environments (step 2 in Figure 1) and to provide localization functions to other robots and even to flying vehicles such as quadcopters (step 3 in Figure 1).

Self-localization without a pre-installed localization infrastructure is particularly challenging but absolutely necessary in the domain of Sensor and Actor Networks (SANETs) [3], [4]. One of the most promising approaches is the Mass-Spring-Relaxation (MSR) self-localization algorithm [5]. The

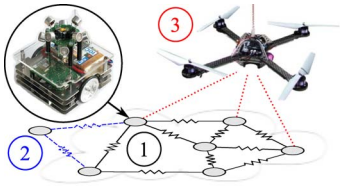

Fig. 1. Autonomous Localization Framework (ALF)

basic idea is to exploit the second law of Newton: Atoms, in our case sensor nodes, are finding their correct position by yielding to forces which are caused by neighbor atoms, until the entire grid becomes stable. MSR is known for its simplicity and fault-tolerance. For example, the approach can compensate systematic measurement errors without propagating those throughout the grid. Howard et al. [5] have been among the first who utilized MSR for solving a localization problem. They use beacons with known (fix) positions, i.e. anchor nodes to initialize the MSR. Thus, one of the biggest disadvantages is that *a priori* knowledge of the network structure is required. Priyantha et al. [6] compensated this by inserting an additional pre-stage where initial position knowledge is obtained. Still, global knowledge is generated by collecting position information at a central node. Without such *a priori* knowledge, the basic MSR approach fails with high probability even for very small network sizes [6]. Besides of scalability and reliability issues, this strategy of generating global state is not suitable for many SANET applications, especially if nodes are mobile (in this case, the initial setup can take a very long time) or if the number of nodes is unknown.

In this paper, we describe an advanced MSR algorithm that is not subject to the mentioned limitations. In particular, we added features that allow a fully distributed and stateless operation. Nodes can arrive or depart from the reference grid at any time. In the context of this paper, we assume a mobile robot platform, which can determine the distances to its direct neighbors and which is able to exchange information among this neighbor set. In general, we rely on our own hardware platform that uses ultrasound distance measurements together with a rough estimate of the angle [7], [8]. As can be seen from our experiments, the advanced MSR algorithm is robust and scalable due to its self-organizing operation principle. Furthermore, it reduces the necessary number of information

978-1-4577-0636-3 /11/$26.00 ©2011 IEEE

exchanges between the nodes to a minimum in order to prolong the network lifetime [9].

The key contributions of this paper can be summarized as follows: We present a fully self-organizing approach for creating and maintaining a self-localization coordinate system (Section III). Our approach is based on the classical MSR algorithm, which we extended to operate in a distributed and fully self-organizing way. We extensively evaluated the localization quality as well as the communication overhead (Section IV) both in a simulation framework and in an experimental lab setup. Compared to related approaches such as the classical MSR and the extensions by Priyantha et al., our approach provides better scalability and robustness even for increasing network sizes.

II. Related Work

A. Localization Approaches

We can classify different localization approaches according to the used techniques. This includes measures of the radio connectivity, the Received Signal Strength Indicator (RSSI), the Angle of Arrival (AoA), the Time of Arrival (ToA), and the Time Difference of Arrival (TDoA). Some works are using a combination of multiple techniques. One of the most frequently used distance estimation between any two nodes is the RSSI, because it is practically always available. However, it is unstable in most circumstances and varies over time [10]. Furthermore, we can classify the self-localization schemes w.r.t. the necessary infrastructure, i.e. the use of anchor nodes. These anchors, which are also called beacons, *a priori* know their exact positions that are used to update other nodes' positions [11], [12]. In general, centralized algorithms are very accurate in static networks, but need to collect all relevant information at a central point to calculate the relative positions of each node. In distributed versions, each node has to assign itself a position based on information received from neighboring nodes [6], [12]. Such approaches are typically more fault-tolerant, robust, and scalable. They also deal better with dynamic environments at the expense of localization accurateness.

We focus on distributed algorithms, thus, anticipating greatly simplified self-localization in very dynamic environments. We also consider the ability that the nodes can join the network in any order and at any time. Therefore, solutions requiring a global initialization phase are not suitable such as the concept presented by Priyantha et al. [6]. Approximate node positions may be generated first, before starting the localization process.

An important set of recent approaches can be categorized as patch-and-stitch [13]. The network is represented by a graph and divided into a set of sub-regions that create local maps, i.e. *patches*, that are *stitched* in several steps to create the final graph. This type of technique is sensible to the so-called *flip error* (parts of the graph are overlapping), when nodes join the local maps. It can propagate the error into the local map stitching process. This phenomenon is also called *flip ambiguity*. It is in general a major problem in lateration based WSN localization that introduces large positioning errors. Usually a lot of computational effort is needed to detect and reduce this effect [14]. Another problem related with the patch-and-stitch technique is the existence of orphan nodes, which cannot be integrated in the localization system [15]. Other approaches like hop-count based schemes, e.g. [16], or anchor-based algorithms, e.g. [11], are extremely sensitive to node density and anchor distribution, respectively. Therefore, the accuracy is limited in randomly deployed SANETs.

B. MSR-based Self-Localization

The starting point for our self-localization algorithm is the Mass-Spring-Relaxation (MSR) algorithm [5]. The basic model is similar to a molecule grid structure. As the name indicates, mass points m are connected with linear-elastic springs to each other. The lengths of the individual springs are proportional to the real distances of the mass points. According to Hooke's law, the resulting force $\overrightarrow{F}_j$ affecting a mass point m_j can be computed as:

$$\overrightarrow{F}_j = \sum_i \overrightarrow{F}_{j,i} = \sum_i -\overrightarrow{e}_{j,i} k_{j,i} \Delta l_{j,i} \tag{1}$$

It is depending on its current position $\overrightarrow{x}_{t,j}$ in relation to its interfering i neighbors. $\overrightarrow{e}_{j,i}$ represents the n-dimensional unit vector in the direction from mass point m_j to point m_i. The scalar constant $k_{j,i}$ represents the spring constant for the involved springs, and the scalar value $\Delta l_{j,i} = l_{j,i} - l^0_{j,i}$ represents the length variation/defect of the spring.

If $\overrightarrow{F}_j$ is a non-zero vector, the mass point m_j is within an unstable state. MSR, according to its counterpart in Nature, tries to minimize the energy level of the mass point m_j by yielding to the force. Obviously, this can only be done by moving the mass point m_j to an equilibrium location $\overrightarrow{x}_{equ(t),j}$. It is defined as a place where the sum of all participating forces is zero $\|\overrightarrow{F}_j\| = 0$. Please note that this state may be met while the individual terms $\overrightarrow{F}_{j,i}$ of the sum in Equation 1 are non-zero. If this happens, a local optimum is reached. The mass point m_j has reached an equilibrium state, i.e. it stops to move, but is not relaxed, because individual forces still exist. A relaxed state is defined were each spring found its correct length, i.e. $\|\overrightarrow{F}_{j,i}\|_{\infty,i} = 0$. However, as long as $\overrightarrow{F}_j$ is non-zero, the mass point m_j is moving according to the second law of Newton. Exploiting this and changing from continuous to discrete time, position updates over time can be computed as:

$$\overrightarrow{x}_{t+1,j} = \frac{1}{2m_j} \overrightarrow{F}_j \Delta t^2 + \overrightarrow{x}_{t,j} \tag{2}$$

Howard et al. [5] use a set of beacons assisting mobile robots in the self-localization process using a mass-spring based optimization model. They exploited two features of their test hardware to solve the problem of equilibrium states in a centralized way: reference points, i.e. anchor nodes, and initial node coordinates. Thus, the approach strongly depends on accurate initial coordinate assignments; otherwise the algorithm fails. Priyantha et al. [6] addressed this problem by creating

an intermediary layer between the distance measurements and the solving phase. Within this new phase approximate initial positions are generated. Basically, all nodes have first to take all measurements and to collectively generate rough initial positions, before starting the MSR. This requires synchronization among all the nodes and denies spontaneously arriving nodes. Le et al. [12] use static anchor nodes to optimize the procedure.

In this paper, we provide several extensions to the classical MSR algorithm to allow a continuous setup of the grid without any *a priori* knowledge. To the best of our knowledge no distributed MSR based algorithm exists which is scalable and does not require any kind of additional (global valid) information, knowledge or synchronization. In the scope of this paper, we only discuss 2D localization, however, it can be extended to 3D without significantly increased complexity.

III. Distributed Mass-Spring-Relaxation

Our advanced MSR algorithm is using three key features, which are only based on distance measurements. Additionally, an optional feature can be used if the hardware allows to determine the Angle of Arrival (AoA) at a granularity of at least two sectors ($\pm 90°$ accuracy). For our algorithm, we assume that no global knowledge is available and there is no global synchronization among the nodes. Thus, all decisions must be done in a distributed, self-organized way. On the one hand, this results in a more complex information flow. On the other hand, a stateless behavior can be achieved, i.e. nodes are allowed to arrive to or depart from the network at any point in time.

We assume that the network is represented by an underlaying connected graph $G = (V, E)$ with vertex set V and edge set E. Node j is associated with the vertex $N_j \in V$ and represents a mass point m_j in the physical model. Each node is aware of its direct neighbors. A neighbor is defined as a node that is in range of the distance measurement hardware and both nodes can directly communicate with each other. Such a node pair (i, j) is represented by an edge $(N_i, N_j) \in E$ in the graph G (i.e., a spring in the physical model).

Two different mapping representations $p : V \rightarrow \Re^2$ for the underlying graph G are required. The representation p_r is called *ground truth* and defines the real (unknown) physical coordinate system. It assigns a $(x, y)^T$ position tuple to a node. The real distance from node N_j to its neighbor N_i is represented in the model as the desired length $l^0_{j,i}$ of the connecting spring. For a correct graph representation $l^0_{j,i} = \|p_r(N_j) - p_r(N_i)\|$ must hold for all vertex pairs $N_j, N_i \in V$. This inter-node distance $l^0_{i,j}$ needs to be measured only once or is presented as a slowly moving average of continuous measurements to reduce the jitter. Due to the lack of reference points, the algorithm cannot reconstruct the representation p_r correctly. However it generates a representation p_v that is called *virtual coordinate system* by trying to optimize the following equations for all vertex pairs $N_a, N_b \in V$ so that s is minimal, i.e. $s = min(\sum_{a,b} \epsilon_{a,b})$:

$$\|p_r(N_a) - p_r(N_b)\| = \|p_v(N_a) - p_v(N_b)\| + \epsilon_{a,b} \quad (3)$$

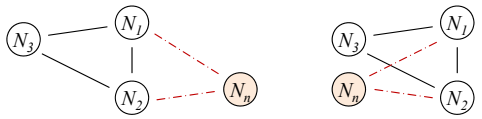

Fig. 2. Ground truth (left) and virtual coordinate system (right)

Consequently, the virtual spring length in the model is defined as $l_{i,j} = \|p_v(N_j) - p_v(N_i)\|$. The physical model operates on the virtual coordinates, therefore $\overrightarrow{x}_{t,j} \equiv p_v(N_j)$.

A. Helper Nodes

One of the main problems is depicted in Figure 2. We assume nodes N_1 to N_3 have been accurately placed and node N_n just joined the network. The figure shows two solutions for the same network: On the left, ground truth is outlined, which also represents one solution according to the distance measurements. On the right, a second possible solution is depicted. Node N_n has a connectivity-degree of two. Therefore, two relaxed positions are possible. Depending on the initial position of node N_n, it can reach both states with a probability of 50 %. Node N_n of Figure 2 represents a sparsely connected branch of the constructed network which is folded/flipped in. This represents a major and frequent problem in most localization techniques which is not restricted to sparsely connected branches. In general, two reasons exist for such a local optimum: First, the network graph is not rigid, i.e. more than one relaxed position exists. Secondly, the algorithm reaches an equilibrium but not a relaxed state.

Nodes in such a local optimum can neither detect nor solve this problem without invoking additional nodes. Constant forces on a node (like within an equilibrium state) are not a reliable criterion. However, a helper node N_h (in our example, this could be either N_1 or N_2) located on the folding map could detect this issue without multi-hop communication. Therefore, we extended the aggregated neighbor table: For each neighboring node, all the connected neighbors are stored as well. Note, no positions or distances are required, just the identification. Figure 3 depicts the available extended neighbor information in a tree notation for node N_1. The maximum tree depth of two has already been reached. Positions and distances are available and stored for the nodes on the branches (direct neighbors) but not for the nodes on the leafs (two-hop neighbors). For the local detection of misplaced nodes, the following two propositions are required: Equation 4 indicates whether two nodes are in distance measuring range l^0_{max} according to their current virtual positions; Equation 5 reflects the direct connectivity between two nodes:

$$M(N_a, N_b) \quad : \quad \|\overrightarrow{x}_{t,a} - \overrightarrow{x}_{t,b}\| \leq l^0_{max} \quad (4)$$

$$D(N_a, N_b) \quad : \quad (N_a, N_b) \in E \quad (5)$$

A helper node N_h can consequently conclude whether two neighboring nodes (in our example, nodes N_n and N_3) are within a local optimum if the condition $C_{a,b}$ is met:

$$C_{a,b} = M(N_a, N_b) \wedge \neg D(N_a, N_b) \quad (6)$$

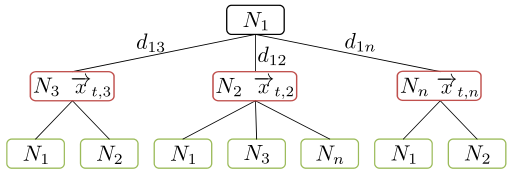

Fig. 3. Sample local information tree for node N_1 (see Figure 2)

If two nodes N_3 and N_n are virtually in close range (Equation 4) but are not connected (inverse of Equation 5) (see Figure 2 right side) then the actual physical distance between the nodes $\|p_r(N_3) - p_r(N_n)\|$ must be greater than the maximum measurements distance l^0_{max} (see Figure 2 left side). Otherwise Equation 5 would be met. Therefore a local optimum is detected.

Algorithm 1 depicts the resulting solution. If a local optimum is detected, the algorithm choses one of the involved neighboring nodes to be re-positioned. The correct node cannot be clearly identified based on live-time, connectivity, or current forces of the participating nodes, thus, the correct node is chosen with a probability of $P_C = 50\%$. If the wrong node gets chosen the local area will get destabilized. Until it gets stable again the region is highly vulnerable and might collapse. In the following section, we will show how the probability P_C can significantly be increased.

According to our experiments, the algorithm works best if the misplaced node gets a new position far away from its old one in order to tear its neighbors into other positions, too. If the new position is too close to the old one, the neighbors would be pulling it back again. We call the re-positioning *jumping*, because from a visual point of view, a node instantaneously performs a large position change.

In several experiments, we figured out that executing the algorithm for each relaxation step is not feasible. Too many jumps of correctly placed nodes are taking place and might be further re-positioned by another helper node – leading to a collapse of the entire system. Furthermore, only one misplaced node should be jumped each time in a local context to prevent local destabilization. We received astonishingly good results if this algorithm is executed only with a low probability of about $1\% \leq P_h \leq 10\%$ after a relaxation step and if it is moving at most one node. All other participating nodes have to reach at least an equilibrium state $V_E \subseteq V$. This equilibrium is defined analog to the relaxed state. If the euclidean norm of the effecting force $\overrightarrow{F}_j$ of a node N_j is less or equal to the threshold θ (see next section), the node N_j is within an equilibriums state: $\|\overrightarrow{F}_j\| \leq \theta \rightarrow N_j \in V_E$.

Algorithm 1 Helper Node Decision ($N_h \in V_E$)

while $\exists N_a, N_b \,|N_a \neq N_b; (N_a, N_b) \in E; N_a, N_b \in V_E$ **do**
  if $C_{a,b} =$ **true then**
    Node $N_r \leftarrow$ randomlyChoose(N_a, N_b)
    $p_v(N_r) \leftarrow$ pointReflectNodeToHelper($p_v(N_r)$)
    $V_O := V_O \cup N_r$
  end if
end while

B. Virtual Anchor Nodes

Anchor nodes strongly improve the quality and speed of the self-localization process: They stabilize the whole system and correlate the real and the virtual coordinate system. In MSR, the mass of anchors is set to infinity ($m_a = \infty$). In our self-organizing system, we have no fixed anchor nodes. However, we can use specific nodes as *virtual anchors*: When a node is in a stable and relaxed position for a long time period, it can be promoted to become a virtual anchor $V_A \subseteq V$ by increasing its weight. Obviously, the mass of such a node should not infinitely be increased – promotion might also happen if the node is misplaced.

We experimented with the following parameters: A node's weight $m_{t,j}$ can be either 1 kg (normal node) or 4 kg (anchor node $\rightarrow$ moving much slower); the initial weight of a node being 1 kg. A threshold θ for the maximum of the individual forces $\|\overrightarrow{F}_{j,i}\|_{\infty,i}$ indicates whether the node N_j is relaxed, i.e. $\|\overrightarrow{F}_{j,i}\|_{\infty,i} \leq \theta \rightarrow N_j \in V_R \subseteq V$, or not, i.e. $\|\overrightarrow{F}_{j,i}\|_{\infty,i} > \theta \rightarrow N_j \notin V_R$. The value θ is to be chosen such that hardware-specific distance measurement errors result in lower forces as θ (noise canceling). In our experiments, we assumed a cumulative spring constant of $k_i = 1Nm^{-1}$ (for simplicity). Therefore, we set θ = 0.1 N (twice of our maximum measurement error range $\zeta = 5cm$ ($\theta = 2\zeta k_i$)) to safely cancel out measurement uncertainties. A semaphore for each node is increased linearly to switch between normal and anchor state at certain thresholds (ρ_1, ρ_2). The following equation must hold: $\rho_1 \geq \frac{2}{P_h}$, P_h being the probability to execute the helper node algorithm. Therefore a node gets checked for a local optimum at least once (with a very high probability) by a neighbor before it can become a virtual anchor. ρ_2 strongly depends on the utilized physical values. In consequence, this provides a fair trade-off between stability, execution time, and energy consumption. The resulting system is depicted in Algorithm 2.

As stated previously virtual anchors have additionally the potential to rise the probability P_C of selecting the correct

Algorithm 2 Anchor Node Decision for N_j

static anchorState $\leftarrow$ initialState
if $N_j \notin V_A$; $\forall N_i \in V_E$; $N_j \in V_R$;
$\forall C_{i,j} \neq$ **true** | $(N_j, N_i) \in E$; anchorState $< \rho_1$ **then**
  anchorState $\leftarrow$ anchorState + 1
else if $N_j \in V_A$; $N_j \notin V_R$; anchorState $> \rho_2$ **then**
  anchorState $\leftarrow$ anchorState - $\|\overrightarrow{F}_{j,i}\|_{\infty,i}\ N^{-1}$
end if
if anchorState $\geq \rho_1$; $N_j \notin V_A$ **then**
  $V_A := V_A \cup N_j$
else if anchorState $\leq \rho_2$; $N_j \in V_A$ **then**
  $V_A := V_A \setminus N_j$
end if

node during a node jumping decision (see section III-A) to nearly 100 % by adding a single condition: Nodes may only be jumped if their weight is close to the lower bound. The network is growing slowly over time and nodes become anchors. New, potentially flipped, nodes get always initialized as non-anchors. If those become candidates in a node jumping process, there is a high probability that the other candidate is a well placed anchor and therefore cannot be selected for jumping.

C. Distributed Initial Phase

The quality of MSR strongly depends on the quality of the initial positions of all the nodes. In contrast to Priyantha et al. [6], we do not rely on an initial position approximation using synchronized nodes, but we use the existing self-localization grid for this purpose. The grid is not constructed all at once but it gets stepwise extended.

Whenever a node N_j wants to join a network V it has to find an attractive position in advance. The conditions for finding such a place are mainly scenario driven and therefore out of scope of this paper. One requirement to the network itself must additionally be fulfilled. It is important to avoid that the candidate N_j joins in to an unstable area of the network. Each neighboring node $N_i \in V^{i,j}$ in connection range (Equation 8) must be relaxed:

$$V^{i,j} \setminus V_R = \emptyset \tag{7}$$

$$V^{i,j} := \{\forall N_k \in V \mid \|p_r(N_k) - p_r(N_j)\| < l^0_{max}\} \tag{8}$$

In order to create accurate initial positions at least $d^+_{\tilde{G}}(N_j) \geq 1$; $\tilde{G} = (V_A \cup \{N_j\}, E)$ connected virtual anchors should also be available. The initial position – after passing all the scenario and advanced MSR driven conditions – is computed according to a lateration technique described in [8]. The neighboring nodes $N_i \in V^{i,j}$ serve as reference points for the computation. Thereby the forces affecting the newly initialized node N_j and, *vice versa*, the network are kept minimal.

D. Sector of Arrival

The three presented extensions to MSR are purely based on distance measurements. However, due to the lack of omni-directional sensing capabilities of single sensors, many distance-ranging devices are build upon several sub-sensors. For example, our ultrasound based hardware has a sensing array of four ultrasound sensors to cover a hemisphere-like detection range [8]. Thus, the AoA can be determined, or, due to imprecisions, at least the Sector of Arrival (SoA). We will show that using only four sectors significantly improves the generation speed and accuracy of the system.

In a first step, the virtual heading α_j of node N_j is estimated based on the virtual coordinates $p_v(\cdot)$ of its neighbors and the corresponding SoAs. Using this additional information, the number of required reference points for the initialization phase can be reduced by one, i.e. only two (instead of three) neighbor connections are required to determine a unique position. Furthermore, the problem of jumping nodes can be handled much more efficiently. Following the concept of helper

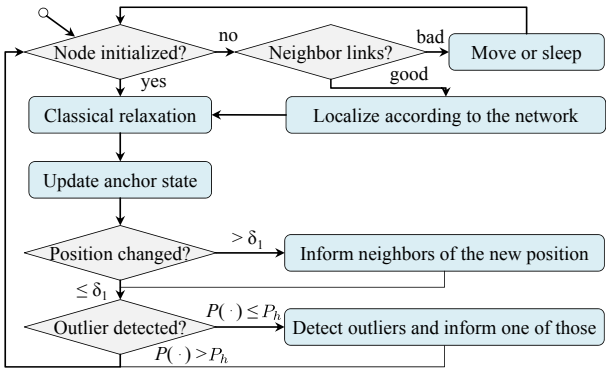

Fig. 4. Flowchart of the advanced Mass-Spring-Relaxation

nodes, the helper N_h is checking whether all of its neighbors $\forall N_i;\ (N_h, N_i) \in E$ are within their predicted SoA $\sigma_{h,i}$. If a distance measurement is not within its theoretically assigned sector, it is marked as an outlier. This condition can be further weakened depending on the accuracy of the used hardware: It might be sufficient to be within the assigned or flanking sectors. A new position $p_v(N_i)$ for a possibly misplaced node N_i is computed by the detected angle, the measured distance and the virtual position $p_v(N_h)$ of the helper node N_h ($\Phi(\alpha, d)$ converts polar into cartesian coordinates):

$$p_v(N_i) \leftarrow \Phi(\sigma_{h,i} + \alpha_h, \|p_r(N_i) - p_r(N_h)\|) + p_v(N_h) \tag{9}$$

E. Complete Algorithm

The resulting algorithm is outlined in Figure 4, which briefly summarizes the main local processing loop of a node. Before the loop starts, we assume that the node already prepared a list of all direct neighbors, the corresponding distances and positions, and the 2-hop-neighbors. The main loop starts by checking whether the node is part of a network. If not, it tries to initialize itself by finding a connection to an existing grid (Section III-C). The second step is the normal relaxation step according to Equation 2. After checking the virtual anchor state (Section III-B), the new position (if there is a change) is transmitted to the connected neighbors. The main loop ends with outlier detection (Sections III-A and III-D). If possible outliers are detected, one is elected and jumped.

N.B.: for practical implementation is it obvious that a robot needs to reset if no convergence is reached for a long time.

IV. Performance Analysis

A. Test Cases and Metrics

We incrementally evaluated our advanced MSR in comparison to the classical MSR algorithm using a custom-made JAVA simulator. All the test cases are listed in Table I. We already added some slight improvements to MSR as described in [5] to allow a distributed position estimation: The nodes are moving around and as soon as they find a place with significant connectivity to an existing network, they connect and start the MSR regardless in which state those neighbors are. The initial positions are generated randomly.

A frequently used metric is known as Global Energy Ratio (GER) [6]. Basically, a normalized error $\widehat{e}_{ij}$ between nodes N_i and N_j is computed (the distance $d_{ij} = \|p_r(N_j) - p_r(N_i)\|$ represents the ground truth and $\widehat{d}_{ij} = \|p_v(N_j) - p_v(N_i)\|$ the result of the algorithm): $\widehat{e}_{ij} = \frac{\widehat{d}_{ij} - d_{ij}}{d_{ij}}$. This error is afterwards used for computing the GER (N represents the network size $|V|$): $GER = \frac{\sqrt{\sum_{i,j:i<j} \widehat{e}_{ij}^2}}{N(N-1)\frac{1}{2}}$. Unfortunately this metric is not independent of the network size: errors are getting smaller with increasing size. A slight modification is the Root Mean Square (RMS) of the normalized error: $RMS = \sqrt{\frac{\sum_{i,j:i<j} \widehat{e}_{ij}^2}{N(N-1)\frac{1}{2}}}$. This value is independent of the network size and larger errors have a much higher impact.

B. Self-localization Quality

The performance of self-localization algorithms strongly depends on the network size, i.e. the resulting errors usually increase with the network size. Thus, we simulated different network sizes of 2 to 512 nodes to evaluate this effect. We generated 200 random topologies for each network size, thus, we executed 7200 runs. In the following figures, we plot the median (50 % quantile) of the measurement samples. The error bars indicate the stable co-domain (25 % and 75 % quantiles).

In a first experiment, we analyzed the node connectivity for all test cases that strongly influences the success rate. Higher connectivity means higher reliability but also higher system costs. In the literature, typically 8 to 15 neighbor connections per node have been used for sufficiently accurate self-localization. This high number, however, results in a significant cost overhead. In our system, a node is only allowed to stay in a certain area if its connectivity degree is less or equal to 5 – this results in a maximum connectivity of 6. As can be seen in Figure 5, all algorithms are becoming quickly saturated as the node size increases. Because the same connectivity condition is in place, only slight variations can be seen.

Figure 6 shows the RMS of the normalized error between each node permutation. As can be seen, the original MSR performs very poor for increasing network sizes. The figure also shows that introducing proper initial positions and checking the state of neighboring nodes (MSR+init) constantly improves the solution, yet, networks containing more than six nodes can still not properly be solved. The introduction of helper nodes (MSR+helper) improves the system: Networks of up to 20 nodes can be solved with high accuracy. Beyond this size, the performance degrades quickly, because the helper node strategy has only a local impact on the network. In large networks, there is a high probability that a folded area grows before it can be corrected. Virtual anchors (MSR+virtA) counteract this problem even for networks of 64 nodes and more. They bring more stability into the system and the probability P_C for the helper node to select the correct node for jumping is significantly improved. In very large networks, significant errors can still be observed. Finally, the full version of our advanced MSR (advMSR) additionally uses the SoA. It performs very well for all simulated network sizes.

TABLE I
TEST CASES

Algorithm	Description
MSR	classical MSR according to [5]
MSR+init	distributed initial positions (Section III-C)
MSR+helper	additional helper nodes (Section III-A)
MSR+virtA	use of virtual anchor nodes (Section III-B)
advMSR	full version of our advanced MSR algorithm

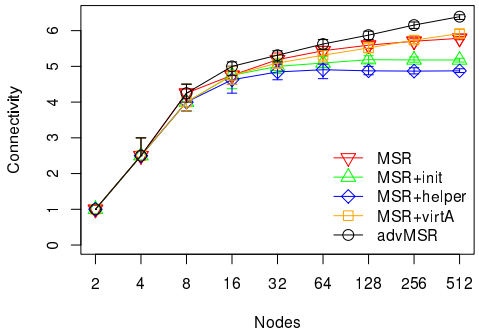

Fig. 5. Self-localization accuracy: connectivity degree

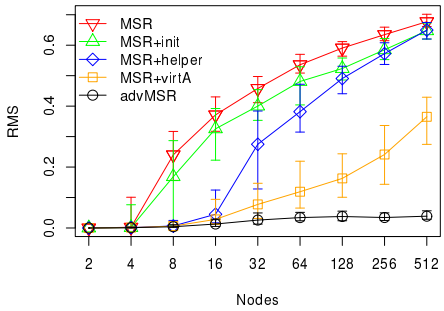

Fig. 6. Self-localization accuracy: RMS

C. Influence of Communication Aspects

In the centralized MSR, all the measurements need to be collected at a single sink, thus, generating a high traffic load towards this sink. In our distributed scenario, each optimization step requires the exchange of information among neighboring nodes. In the following, we analyze the network traffic necessary for the advanced MSR.

We assume that a node can simultaneously inform all of its connected neighbors using only a single broadcast packet. We investigated four network sizes and stopped the simulation run as soon as a *sufficient accuracy* has been reached. Figure 7 (left) shows the number of packets until convergence (we call this *continuous mode*). The graph is shown in form of a boxplot: For each data set, a box is drawn from the 25 % to the 75 % quantile, whiskers for the 2.5 % and 97.5 % quantile, and the median is marked with a thick line. Data points outside the statistical evaluation are considered outliers and drawn separately. On average, a node has to send 250 packets, which is a substantial overhead. Please note that we used the global knowledge of the simulator to determine that every node has

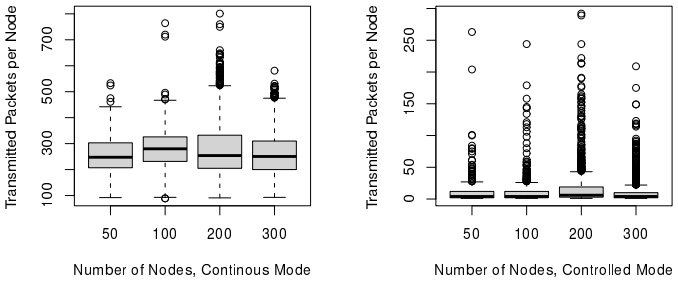

Fig. 7. Transmission frequency

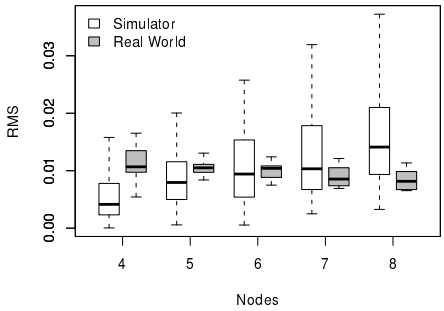

Fig. 9. Comparison of simulation and experimental results

reached convergence and stopped the simulations at that point. This strategy is not suitable for a real application.

Therefore, we introduced a threshold $\delta_1 = \|\overrightarrow{x}_{t,j} - \overrightarrow{x}_{t-1,j}\| \geq \zeta$ to evaluate the position change since the last packet transmission, which must be reached before sending any new updates (see also Figure 4). This solution is named *controlled mode*. We set $\delta_1 = \zeta$ to cancel out the measurement noise. Using this approach, far less packets are required to get a stable solution as depicted in Figure 7 (right). On average, a node only sends about 10 packets.

Note that in practice a *sufficient accuracy* for the *continuous mode* can not be achieved due to the lack of global knowledge. Each node would continuously transmit almost the same position information using the maximum bandwidth. For the *controlled mode* the nodes automatically stop transmitting as they reach position convergence. Thus, redundant information transmissions are avoided and the main loop in Figure 4 can be paused if no packets were received or transmitted for a long time. But it needs to be resumed if new packet arrive.

We also validated the resulting localization accuracy. Figure 8 shows the quality (RMS) as well as the required transmitted packets per node over the transmission threshold $\delta_1 \in [0; 4\zeta]$ using 32 nodes. As previously, we plot the median of the measurement samples. The error bars indicate the stable co-domain (25 % and 75 % quantiles). It can be seen that the accuracy slightly decreased for higher thresholds δ_1. However, the resulting significant reduction of the required transmissions and the fully self-organized termination clearly motivate the introduction of the threshold δ_1 (*controlled mode* operation).

N.B.: Figure 6 looks similar for both modes ($\delta_1 = \zeta$).

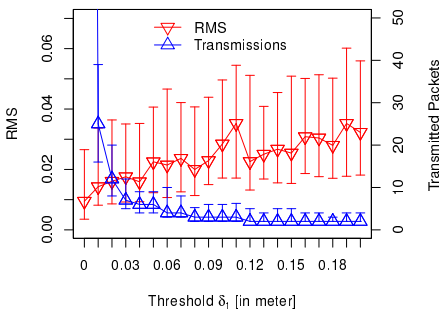

Fig. 8. Self-localization accuracy: evolution of RMS and transmitted packets for varied transmission thresholds δ_1

D. Real World Tests

In order to validate our simulation results, we conducted several real word experiments in our lab. We currently have 8 mobile platforms equipped with our ranging hardware [17] available for experiments. The RMS metric requires accurate information about the ground truth p_r (in particular, distances $d_{i,j};\ \forall N_i, N_j \in V$). Due to the lack of an additional (and more precise) localization system – a yardstick is way too inaccurate for exact evaluation – we were not able to derive these distances outside the system. Therefore, we changed the parameters in a way (basically, distances were reduced), so that each platform itself can measure the distance to all other nodes. This is very accurate based on the ultrasound measurements. Systematic errors of the hardware, like uncalibrated ultrasound devices (due to temperature drifts), are canceled out by this method. Of course, this approach only allows to verify the algorithm but not the hardware components. Only a subset of the edges (distances $\tilde{d}_{i,j};\ \forall (N_i, N_j) \in E;\ |\tilde{d}| < |d|$) of this fully connected graph is then presented to the algorithm.

In general, we used the same parameters for the lab experiment as for the simulator. The results are depicted in form of boxplots in Figure 9, using the same notation as for Figure 7. We tested 5 different network sizes. The connectivity plot (data not shown) is nearly identical to the plot depicted in Figure 5. The co-domain of the lab experiments are always within the co-domain of the simulations.

The maximum difference between the experimental and simulation results at the median is 6.5×10^{-3} (0.65 %). Converted into a length specification, this corresponds to a maximum median placement error of approx. 8 mm (average node distance is 1.25 m), which is below our measurement accuracy. We can conclude that the simulator reflects the behavior of the real system very well.

V. Conclusion

In this paper, we discussed the need for fully distributed self-localization techniques in the context of SANETs. Our application scenario is a group of mobile robots exploring potentially unknown environments. This group forms a so called reference grid, which can then be used for localizing additional systems such as other robots or quadcopters. We extended MSR algorithm to become fully self-organized and to achieve higher localization accuracy and robustness. As

can be seen from our simulation results, our advanced MSR is much better suited for increasing network sizes, without restrictions on anchor nodes or initial phases typically used in related approaches. First tests in a lab environment using real robot systems have been performed to validate the simulation results. Future work include the introduction of obstacles, i.e. the treatment of Non Line of Sight scenarios.

REFERENCES

[1] L. Hu and D. Evans, "Localization for Mobile Sensor Networks," in *10th ACM International Conference on Mobile Computing and Networking (MobiCom 2004)*. Philadelphia, PA: ACM, September 2004, pp. 45–57.

[2] M. L. Sichitiu and V. Ramadurai, "Localization of Wireless Sensor Networks with a Mobile Beacon," in *1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2004)*. Fort Lauderdale, FL: IEEE, October 2004, pp. 174–183.

[3] N. B. Priyantha, H. Balakrishnan, E. D. Demaine, and S. Teller, "Mobile-Assisted Localization in Wireless Sensor Networks," in *24th IEEE Conference on Computer Communications (INFOCOM 2005)*, Miami, FL, March 2005, pp. 172–183.

[4] Z. Zhu, A. Man-Cho So, and Y. Ye, "Universal Rigidity: Towards Accurate and Efficient Localization of Wireless Networks," in *29th IEEE Conference on Computer Communications (INFOCOM 2010)*. San Diego, CA: IEEE, March 2010.

[5] A. Howard, M. J. Mataric, and G. Sukhatme, "Relaxation on a Mesh: a Formalism for Generalized Localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, Maui, HI, October 2001, pp. 1055–1060.

[6] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-Free Distributed Localization in Sensor Networks," MIT Laboratory for Computer Science, Tech. Rep. TR-892, April 2003.

[7] J. Eckert, F. Dressler, and R. German, "Real-time Indoor Localization Support for Four-rotor Flying Robots using Sensor Nodes," in *IEEE International Workshop on Robotic and Sensors Environments (ROSE 2009)*. Lecco, Italy: IEEE, November 2009, pp. 23–28.

[8] J. Eckert, R. German, and F. Dressler, "An Indoor Localization Framework for Four-rotor Flying Robots Using Low-power Sensor Nodes," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 2, pp. 336–344, February 2011.

[9] I. Dietrich and F. Dressler, "On the Lifetime of Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, pp. 1–39, February 2009.

[10] M. Dominguez-Duran, D. Claros, C. Urdiales, and F. Coslado, "Dynamic calibration and zero configuration positioning system for WSN," in *14th IEEE Mediterranean Electrotechnical Conference (MELECON 2008)*. Ajaccio, France: IEEE, May 2008, pp. 145–150.

[11] F. Mourad, H. Snoussi, F. Abdallah, and C. Richard, "Anchor-Based Localization via Interval Analysis for Mobile Ad-Hoc Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 57, no. 8, pp. 3226–3239, July 2009.

[12] V.-D. Le, V.-H. Dang, S. Lee, and S.-H. Lee, "Distributed localization in wireless sensor networks based on force-vectors," in *International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP 2008)*. Sydney, Australia: IEEE, December 2008, pp. 31–36.

[13] O.-H. Kwon, H.-J. Song, and S. Park, "The Effects of Stitching Orders in Patch-and-Stitch WSN Localiization Algorithms," *IEEE Transaction on Parallel and Distributed Systems*, vol. 20, no. 9, pp. 1380–1391, September 2009.

[14] A. A. Kannan, B. Fidan, and G. Mao, "Analysis of Flip Ambiguities for Robust Sensor Network Localization," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 4, pp. 2057–2070, May 2010.

[15] J. Bachrach and C. Taylor, "Localization in Sensor Networks," in *Handbook of Sensor Networks: Algorithms and Architectures*, I. Stojmenovic, Ed. John Wiley & Sons, September 2005, pp. 277–310.

[16] D. Tran and T. Nguyen, "Localization In Wireless Sensor Networks Based on Support Vector Machines," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 7, pp. 981–994, July 2008.

[17] J. Eckert, K. Koeker, P. Caliebe, F. Dressler, and R. German, "Self-localization Capable Mobile Sensor Nodes," in *IEEE International Conference on Technologies for Practical Robot Applications (TePRA 2009)*. Woburn, MA: IEEE, November 2009, pp. 224–229.