Comparison of Message Passing Algorithms for Cooperative Localization under NLOS Conditions

Samuel Van de Velde*, Henk Wymeersch†, and Heidi Steendam*
*Department of Telecommunications and Information Processing,
Ghent University, Belgium, e-mail: {slvdveld, hs}@telin.ugent.be
†Department of Signals and Systems, Chalmers University of Technology,
Gothenburg, Sweden, e-mail: henkw@chalmers.se

***Abstract*—For large wireless networks, there is a need for accurate localization in a distributed manner. Several algorithms have been developed in order to achieve this goal. However, comparing different algorithms is hard because of the use of different network topologies and measurement models. In this paper two promising message passing algorithms, called SPAWN and SLEEP, are compared in terms of accuracy, complexity, and network traffic. To enable a fair comparison, several alterations are made to SLEEP resulting in ASLEEP with reduced network traffic and the incorporation of reference nodes. Simulations, using measurement models from real ultra-wideband equipment, show that ASLEEP is able to achieve similar estimation quality as SPAWN at much lower complexity and network traffic.**

***Index Terms*—Cooperation, localization, ultra-wideband, SPAWN, SLEEP.**

I. Introduction

More and more, modern mobile applications are exploiting position information about the user in order to provide an augmented user experience. Many of these applications have not yet reached their full potential due to some important limitations of the positioning system they rely on, such as GPS or WiFi. At this time, there is no scalable system that can provide accurate position information at low cost for both indoor and outdoor environments. This may change however, with the adoption of ultra-wideband (UWB) technology [1] and cooperative localization [2]. By equipping users with a low cost UWB transceiver, it becomes possible to make precise range measurements necessary for accurate positioning. UWB can be used in a peer-to-peer mode, enabling a cooperative approach where all users help each other in finding and refining their position estimates.

In large networks, centralized cooperative processing is not feasible due to the exponential increase (in the number of nodes) in network traffic [3]. Because of this, considerable attention has been given in developing distributed algorithms, with only linear increase in network traffic. In [4], an iterative algorithm (called SPAWN) based on belief propagation (BP) is presented. With this algorithm, very accurate results are achieved but at very high complexity and large packet requirements. In [5] another iterative algorithm (called SLEEP) uses expectation propagation (EP) [6] in order to reduce complexity and packet size. However, both algorithms have never been compared due to the difference in measurement models and because SLEEP is actually not fully distributed, as it requires

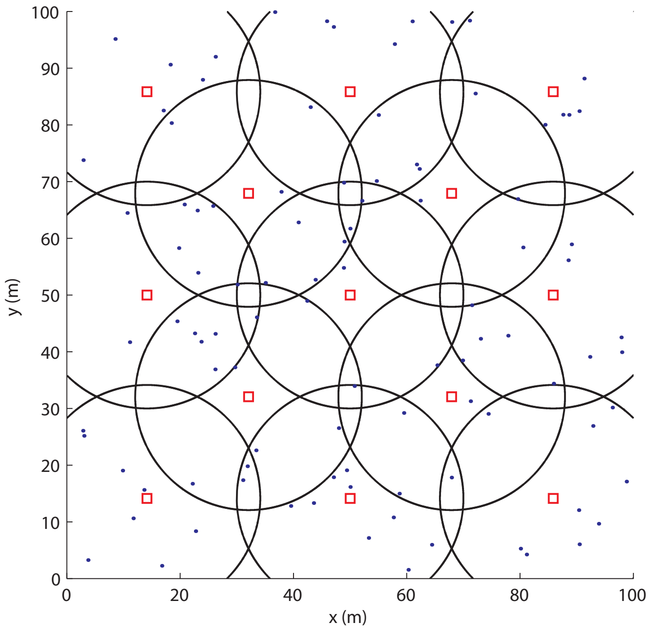

Figure 1. The simulation setup: a 100 m $\times$ 100 m area with 13 reference nodes and a communication range of 20m.

a separate centralized step to provide absolute coordinates. Furthermore, it will be shown that with SLEEP, the network easily becomes flooded with small packets and may cause large delays in localization.

In this paper, two alterations to SLEEP are presented in order to solve some of the issues mentioned above. This altered version of SLEEP, called ASLEEP, is compared to SPAWN for a static scenario in terms of performance, complexity, and network traffic for a number of network topologies (see Fig. 1). Both algorithms are tested using UWB measurements for scenarios with mixed line-of-sight (LOS) and non-line-of-sight (NLOS) conditions.

II. Problem Formulation

A. System Model

We consider a wireless network with N user nodes with unknown position and N_{ref} reference nodes for which the position is known. Every node i is connected to a number of neighboring nodes collected in $\mathcal{N}(i)$ with whom messages can be exchanged and ranging can be performed. The goal of the users is to locally estimate their position $\mathbf{x}_i$ based on the position of the reference nodes and the estimated distances with their neighbors. User node i can estimate the distance to node $j \in \mathcal{N}(i)$, denoted by $z_{j\to i}$, with a ranging protocol, e.g., two-way time of arrival (TW-TOA) [7]. All measurements by all nodes are collected in a vector $\mathbf{z}$. The joint posterior distribution for $\mathbf{x}_{1:N}$ can be written as follows using Bayes' theorem:

$$\begin{aligned} p(\mathbf{x}_{1:N}|\mathbf{z}) &\propto p(\mathbf{z}|\mathbf{x}_{1:N})p(\mathbf{x}_{1:N}) \\ &= \prod_{i=1}^{N+N_{\mathrm{ref}}} p(\mathbf{x}_i) \prod_{j\in\mathcal{N}(i)} p(z_{j\to i}|\mathbf{x}_i,\mathbf{x}_j). \quad (1) \end{aligned}$$

Here, $p(\mathbf{x}_i)$ is the prior probability distribution for the position of node i. For user nodes, this prior is uniform and can be omitted. For reference nodes, the prior is a Dirac distribution $p(\mathbf{x}_i) = \delta(\mathbf{x}_i - \mathbf{x}_i^{\mathrm{ref}})$, with $\mathbf{x}_i^{\mathrm{ref}}$ the exact position of reference node i.

For modeling the error of the distance estimates, we use measurements collected from real ultra-wideband equipment [8]. We shall make a distinction between line-of-sight and non-line-of sight measurements. In LOS cases where there is a direct path between nodes i and j, the measurements exhibit a normal distribution with mean equal to the distance and standard deviation $\sigma = 0.25\mathrm{m}$. The likelihood function for a LOS measurement $z_{j\to i}$ can thus be written as[1]

$$p(z_{j\to i}|\mathbf{x}_i,\mathbf{x}_j) = \mathcal{N}(z_{j\to i}; \|\mathbf{x}_i - \mathbf{x}_j\|, \sigma). \quad (2)$$

In NLOS conditions, caused by signal blockage, the measurements are strictly larger than the true distance and can be modeled according to an exponential distribution with rate parameter $\lambda = 0.38\mathrm{m}^{-1}$. The likelihood function for a NLOS measurement $z_{j\to i}$ can be written as

$$p(z_{j\to i}|\mathbf{x}_i,\mathbf{x}_j) = \begin{cases} \lambda e^{-\lambda|\|\mathbf{x}_i-\mathbf{x}_j\|-z_{j\to i}|} & \text{if } z_{j\to i} \geq \|\mathbf{x}_i - \mathbf{x}_j\| \\ 0 & \text{else.} \end{cases} \quad (3)$$

Throughout the paper, we shall assume that a node is able to identify whether a range measurement is LOS or NLOS, and use the corresponding likelihood accordingly. In [8], [9], several methods are presented that can determine the NLOS or LOS nature of a measurement. We will abbreviate $p(z_{j\to i}|\mathbf{x}_i,\mathbf{x}_j)$ by $f_{j\to i}$.

III. Localization by Message Passing

It has been shown [4], [5] that estimating the position of the users in a distributed manner can be accomplished by using message passing algorithms. With these iterative algorithms, every user is able to find an approximation of the marginal distribution $p(\mathbf{x}_i|\mathbf{z})$ of its position, called the belief $b_i(\mathbf{x}_i)$, by passing messages between neighboring nodes. In the literature, two of the proposed algorithms are SPAWN and SLEEP that respectively use loopy belief propagation (BP) and expectation propagation (EP) as underlying message passing algorithm. These algorithm operate by sending messages over a factor graph [10] that is constructed from the joint posterior distribution (1). A factor graph is constructed by representing all variables of the posterior probability distribution as round vertices, and all factors by square vertices. A square vertex is then connected to all variables that are arguments of its corresponding factor. For the joint posterior distribution (1), two neighbouring user nodes i and j will be represented by two round vertices that are connected by two factors representing the inter-node likelihood functions. This can be seen in Fig. 2. Note that the prior probability distribution is omitted for the two nodes since it is uniform.

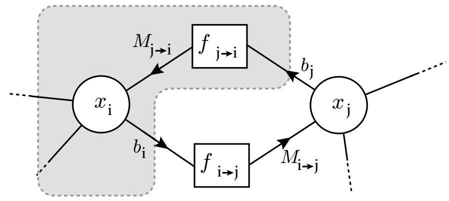

Figure 2. Factor graph representation of the joint posterior distribution (1) for two neighbouring user nodes i and j. The shaded area is the part of the factor graph that can be physically related with node i in the wireless network. Here, $f_{j\to i}$ is shorthand for $p(z_{j\to i}|\mathbf{x}_i,\mathbf{x}_j)$, and arguments of messages are omitted. Message flow is indicated for SPAWN and ASLEEP.

In general, messages are repeatedly sent over the edges of the graph in two directions until all beliefs have converged. However, since nodes are not physically connected with each other, some messages need to be wirelessly transmitted between nodes, which makes message size and overall message count an important factor in comparing these kind of algorithms.

From the literature, it is known that for many problems, BP is able to converge fast to relatively accurate approximations of marginal distributions [11]. However, in case of continuous variables, complexity of the algorithm can become very high and messages can have unbounded complexity [12]. EP is an algorithm developed to address the problems of belief propagation by representing the beliefs by a family of simpler distributions, such as normal distributions [6]. Because of this, messages also reduce to simple distributions and computations become more tractable. However, due to the approximations in EP the quality of the beliefs are expected to be lower than for BP and some convergence issues can arise.

Comparing SPAWN and SLEEP in terms of accuracy, complexity, and message size is thus directly related to the used message passing algorithm. However, the network traffic generated by both algorithms highly depends on the implementation and message schedule of the chosen message passing algorithm.

[1]With $\mathcal{N}(z; m, \sigma)$ representing a normal distribution for the random variable z with mean equal to m and variance σ.

Algorithm 1 SPAWN

for $l = 1$ **to** $N_{\text{iter}}^{\text{BP}}$ **do**
  nodes $i = 1$ **to** N **in parallel**
    Broadcast belief: $b_i^{(l-1)}(\mathbf{x}_i)$
    for $\forall j \in \mathcal{N}(i)$ **do**
      Receive belief $b_i^{(l-1)}(\mathbf{x}_i)$
      Calculate message $M_{j\to i}^{\text{SPAWN}}(\mathbf{x}_i)$ using (5)
    end for
    Update belief $b_i^{(l)}(\mathbf{x}_i) = \prod_{j\in\mathcal{N}(i)} M_{j\to i}^{\text{BP}}(\mathbf{x}_i)$
  end parallel
end for

Algorithm 2 SLEEP

for $l = 1$ **to** $N_{\text{iter}}^{\text{SLEEP}}$ **do**
  select neighbours (i, j) according to schedule
    Request and receive $\frac{b_j(\mathbf{x}_j)}{M_{i\to j}(\mathbf{x}_j)}$ from node i
    Calculate the projection $P_{ij}(\mathbf{x}_i)$ using (6)
    Calculate $b_i^{\text{new}}(\mathbf{x}_i)$ and $M_{j\to i}^{\text{new}}(\mathbf{x}_i)$ using (8) and (9)
    $b_i(\mathbf{x}_i) = b_i^{\text{new}}(\mathbf{x}_i)$ and $M_{j\to i}(\mathbf{x}_i) = M_{j\to i}^{\text{new}}(\mathbf{x}_i)$
  end select
end for

A. SPAWN

In SPAWN, the fact that two independent measurements are available between two user nodes is exploited in order to reduce complexity and network traffic. Because of this, the algorithm works in a synchronized fashion in which every node broadcasts its belief once at the beginning of each iteration. To achieve this, messages are only allowed to flow in one direction and the belief is calculated using only the branches for which a measurement is *locally* available, i.e., for node i, this corresponds to the branches with factor $f_{j\to i}$, $\forall j \in \mathcal{N}(i)$. Because of this, half of the number of branches a node is connected to is used to receive messages, and the other half is used to send messages. Because no message is ever received from an outgoing edge, the message that is sent over an edge is equal to the belief. This makes broadcasting possible. The flow of messages in SPAWN can be seen in Fig. 2. A message received from node j to node i via the factor $f_{j\to i}$ is denoted $M_{j\to i}(\mathbf{x}_i)$. The product of these messages gives the belief of node i [4, line 8 in alg.3]

$$b_i(\mathbf{x}_i) = p(\mathbf{x}_i) \prod_{j\in\mathcal{N}(i)} M_{j\to i}^{\text{BP}}(\mathbf{x}_i). \tag{4}$$

Messages are given by [4, line 9 in alg.3]

$$M_{j\to i}^{\text{SPAWN}}(\mathbf{x}_i) \;=\; \int p(z_{j\to i}|\mathbf{x}_i, \mathbf{x}_j) b_j(\mathbf{x}_j) d\mathbf{x}_j. \tag{5}$$

These messages are calculated *locally* at node i using the received belief of it's neighboring node j and the locally available measurement $z_{j\to i}$. In the beginning of each iteration, every node broadcasts its belief, so that it is available to calculate the message. The complete algorithm is outlined in Algorithm 1. Messages and beliefs are represented with particles as described in [13]. The complexity of (4) scales as $\mathcal{O}(R_1^2)$, where R_1 is the number of particles. The estimated position of each user is obtained by taking the mean of its belief. This corresponds to a minimal mean squared error (MSSE) estimate.

B. SLEEP

SLEEP is a direct implementation of EP for a network where measurements between nodes are shared, i.e., $z_{i\to j} = z_{j\to i}$. Because of this, messages must flow in both directions over the edge between node i and j. In SLEEP, beliefs are again given by the product of all incoming messages. The messages in SLEEP are Gaussian approximations of the messages of BP[2] and because of this, beliefs are also Gaussian. For the approximation of message $M_{j\to i}(\mathbf{x}_i)$, the following projection is performed [5, eq. 2A]

$$P_{j\to i}(\mathbf{x}_i) = \mathcal{P}\left[\frac{b_i(\mathbf{x}_i)}{M_{j\to i}(\mathbf{x}_i)} \underbrace{\int d\mathbf{x}_j p(z_{j\to i}|\mathbf{x}_i, \mathbf{x}_j) \frac{b_j(\mathbf{x}_j)}{M_{i\to j}(\mathbf{x}_j)}}_{\text{BP message}}\right], \tag{6}$$

with $\mathcal{P}[\cdot]$ is the projection operator that is defined as

$$\mathcal{P}[p] = \underset{q\in\mathcal{G}}{\operatorname{argmin}} \operatorname{KL}(p\|q). \tag{7}$$

Here, $\operatorname{KL}(p\|q)$ is the Kullback-Leibler divergence between p and q and $\mathcal{G}$ represents the family of Gaussian distributions. Because the corrected belief $\frac{b_j(\mathbf{x}_j)}{M_{i\to j}(\mathbf{x}_j)}$ is not locally available at node i, it is required for the calculation of (6) that it is transmitted from node j to i. The projection $P_{ij}(\mathbf{x}_i)$ can be seen as a weighted approximation for the standard BP message, i.e., the original BP message towards node i is multiplied with the corrected belief of node i such that the overall approximation would be the most accurate in the region where the belief of node i has a high probability. Because of this approximation however, the algorithm can become unstable. For example, during the first few iterations, when the belief still has a large spread, the approximation that is made can be very inaccurate. The error made by the approximation can then propagate through the network and affect other nodes in subsequent iterations. A common approach to battle this instability is by damping the messages with a factor $\gamma \in [0, 1]$ [14]. Using (6) and damping, beliefs and messages can be calculated as follows [5, eq. 2B-2C]

$$b_i^{\text{new}}(\mathbf{x}_i) \;=\; P_{ij}(\mathbf{x}_i)^{1-\gamma} b_i(\mathbf{x}_i)^{\gamma} \tag{8}$$

$$M_{j\to i}^{\text{new}}(\mathbf{x}_i) \;=\; \left(\frac{P_{ij}(\mathbf{x}_i)}{b_i(\mathbf{x}_i)}\right)^{1-\gamma} M_{j\to i}(\mathbf{x}_i). \tag{9}$$

[2]Note that the message in BP for with shared measurements is different from the message (5) in SPAWN.

Algorithm 3 Altered SLEEP

nodes $i = 1$ **to** N **in parallel**
 Receive position of neighbouring reference nodes
 Calculate $b_i(\mathbf{x}_i) \in \mathcal{G}$ by using SLEEP
end parallel
for $l = 2$ **to** $N_{\text{iter}}^{\text{EP}}$ **do**
 nodes $i = 1$ **to** N **in parallel**
  Broadcast belief: $b_i(\mathbf{x}_i)$
  for $\forall j \in \mathcal{N}(i)$ **do**
   Receive belief $b_j^{(l-1)}(\mathbf{x}_j)$
   Calculate the projection $P_{ij}(\mathbf{x}_i)$ using (10)
   Calculate $b_i^{\text{new}}(\mathbf{x}_i)$ and $M_{j\rightarrow i}^{\text{new}}(\mathbf{x}_i)$ using (8)–(9)
   $b_i(\mathbf{x}_i) = b_i^{\text{new}}(\mathbf{x}_i)$ and $M_{j\rightarrow i}(\mathbf{x}_i) = M_{j\rightarrow i}^{\text{new}}(\mathbf{x}_i)$
  end for
 end parallel
end for

The algorithm is shown in Algorithm 2. Messages are initialized as random Gaussians with a very high spread. With these messages, initial beliefs are calculated using (4). Every iteration, a user will update its belief in an unsynchronized way and transmit a message to one of its neighbours. This is done until a message is sent in both directions over all edges of the factor graph. For networks with relatively few users, the number of neighbours for each node can still be high[3] and as a result, every iteration a large amount of messages will be sent over the network with this algorithm. Furthermore, sharing measurements raises some question on how it is determined which node makes the measurement and how it will be shared. Together with the fact that damped EP generally requires more iterations than BP for convergence, it follows that data traffic with SLEEP is very high. Furthermore, due to the asynchronized update scheme of the nodes, it may take a considerable amount of time before all nodes are updated, resulting in slow positioning.

For two dimensional localization, formula (6) leads to a 4-dimensional integral. By introducing a transformation to new coordinates, this can be reduced to a two dimensional integral that can efficiently be calculated using importance sampling [5]. The operation scales as $\mathcal{O}(R_2)$, where R_2 is the number of samples used for importance sampling.

In the section III-C, an altered synchronized version of SLEEP, called ASLEEP, is presented.

C. Altered SLEEP

When two independent measurements between nodes i and j are available, SLEEP can be transformed to a synchronized algorithm. For this, messages are again restricted to flow in one direction. The projection is now calculated as follows:

[3]For example, in the simulated network as described in Section IV, a user has on average more than 10 neighbouring users.

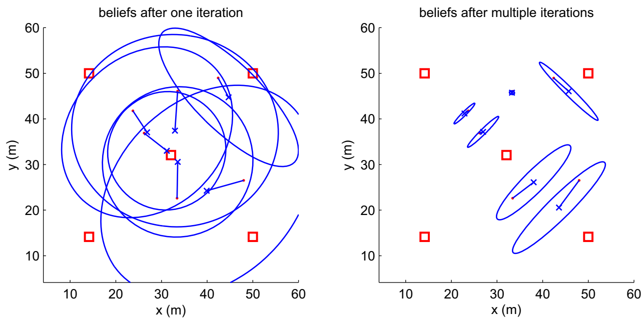

Figure 3. Beliefs of user nodes obtained with SLEEP using only information from reference nodes. Ellipses represent the 95% confidence interval of the belief. Lines represent the error between the mean of the belief and the actual position.

$$P_{j\rightarrow i}(\mathbf{x}_i) = \mathcal{P}\left[\frac{b_i(\mathbf{x}_i)}{M_{j\rightarrow i}(\mathbf{x}_i)} \underbrace{\int d\mathbf{x}_j p(z_{j\rightarrow i}|\mathbf{x}_i, \mathbf{x}_j) b_j(\mathbf{x}_j)}_{\text{message from SPAWN (5)}}\right]. \tag{10}$$

Except for the belief $b_j(\mathbf{x}_j)$ of node j, all necessary information is locally available at node i in order to calculate $P_{j\rightarrow i}(\mathbf{x}_i)$. Because of this, with ASLEEP, beliefs can be broadcast once at the beginning of each iteration. Beliefs and messages can again be calculated using (8) and (9).

To obtain absolute coordinates without requiring a separate step, reference nodes are required. Introducing reference nodes can be done in a straightforward way, by placing a prior on the reference nodes in the same way as is done in SPAWN. This approach however leads to instability and slow convergence. The reason for this is that because of the approximations and damping of EP, the beliefs of the users will still have a large spread after the first iteration[4] (see Fig. 3 on the left). It is therefore better to run SLEEP for every user node internally until convergence using only information of the reference nodes. This will make the approximations for the belief more accurate (see Fig. 3 on the right) and hence more useful for neighbours. Furthermore, by doing this, it becomes possible to lower the damping factor for the rest of the algorithm. The complete algorithm for the altered version of SLEEP is shown in Algorithm 3.

IV. Results

We compare the performance of both algorithms in a cooperative network with 100 users and 13 reference nodes by means of Monte Carlo simulations. The reference nodes are positioned in a way to provide maximum coverage in a $100\,\text{m}\times 100\,\text{m}$ deployment area, user nodes are uniformly distributed in the area. The radio range of network nodes is fixed at $20\,\text{m}$ distance. An example simulation scenario is shown in Fig. 1, in total 40 maps with different user positions are

[4]Note that, during the first iteration, no beliefs of neighbours are available and thus only information from the reference nodes is used.

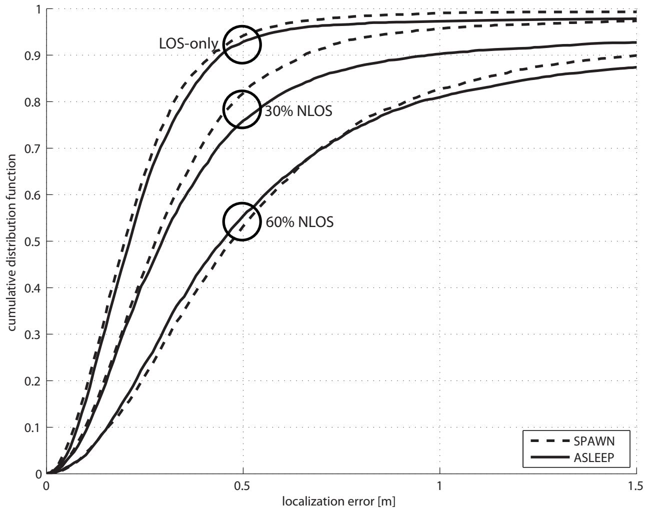

Figure 4. Cumulative distribution function for SPAWN and SLEEP for both LOS and NLOS conditions.

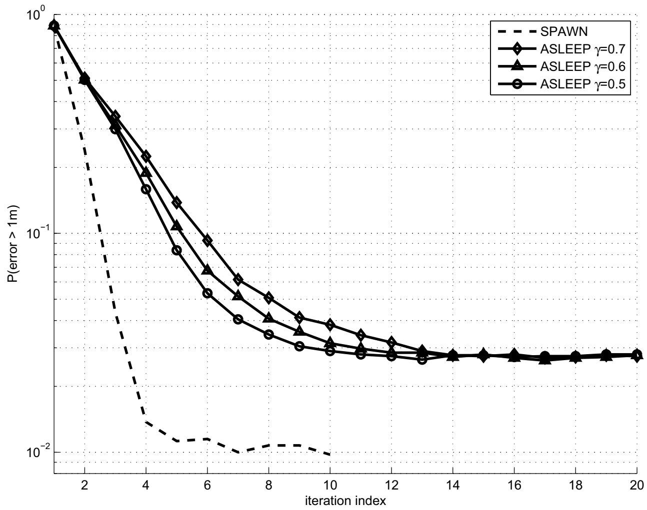

Figure 5. The probability of a positioning error greater than 1 m as a function of the iteration index for a LOS-only scenario.

used for the simulations. Note that, with this network topology, there is on average one user every $100\,\mathrm{m}^2$. For SPAWN, the number of particles for the belief was chosen $R_1 = 2048$. The multiplication step was done using a maximum of 5 messages to create a proposal distribution from which samples were drawn and weighted by the other messages. For SLEEP, the integrals were calculated using importance sampling with the number of particles R_2 adaptively chosen between 2048 and 10240. In order to assure convergence, messages with a negative variance are skipped [15].

Fig. 4 depicts the cumulative distribution function (CDF) for both algorithms after convergence. Three scenarios with a varying percentage of LOS connections between nodes are tested. It can be observed that for both algorithms, the accuracy is highest for LOS scenarios and degrades for an increasing percentage of NLOS connections. In comparing the algorithms, it is observed that SPAWN achieves the best results. For the LOS-only scenario however, the difference with SLEEP is small. For most nodes, the belief can be adequately approximated with a Gaussian. For some nodes however, the actual belief is multi-modal[5] and problems may arise with EP. Ideally, the algorithm would make an approximation for these beliefs that span all modes, but in some cases, the algorithm locks on to a single mode with great certainty. Because of this large localization errors can occur. It is observed that when 30% NLOS connections are introduced, the difference between SPAWN and SLEEP becomes larger. This is because there will be more multi-modal beliefs for these scenarios and because the remaining uni-modal beliefs behave less like a Gaussian. For the scenario with 60% NLOS, the CDF curve of SPAWN falls below the curve for SLEEP for errors between 0 cm and 70 cm. The explanation for this is that the number of particles used in SPAWN are insufficient to accurately represent the messages for this scenario, resulting in a lowered positioning accuracy.

In Fig. 5, the probability of an error greater than 1 m is shown as a function of the iteration index for both algorithms in the LOS scenario. It can be observed that SPAWN converges faster than ASLEEP: after 4 iterations, the probability of locating all users within 1 meter remains unchanged around 99%. For ASLEEP, different values for the damping factor γ are used. Decreasing γ makes the algorithms converge faster, however, lowering this value too much will cause unstable behaviour of the algorithm. In our experiments, $\gamma = 0.5$ was the lowest value where the algorithm remained stable all of the time. Here, the probability of locating a user within 1 meter remains stable after 12 iterations around 97%. For the scenario with 60% NLOS connections, more iterations are required for both algorithms until convergence. More than 80% of all users are located within 1 meter after 5 iterations for SPAWN and 13 iterations for ASLEEP (results not shown).

Finally, we also make a comparison in terms of network traffic for all algorithms. Results for SLEEP are also included. For SLEEP, the minimum damping factor was $\gamma = 0.7$ and the algorithm converged after 12 iterations. The number of data packets that will be physically sent over the network will depend on the maximum number of user bytes per packet. For current UWB equipment, the maximum number of user bytes per packet is 1 kbyte.[6] If we use a 32 bit precision floating point representation for the particles of the beliefs in SPAWN, this requires a total of 8 kb, or equivalently, 8 packets for a message in SPAWN. For both SLEEP and ASLEEP, a message consists of 5 real numbers[7] or 20 bytes, which can easily fit in one packet. So, where SPAWN needs to transmit a number of packets that are fully packed with

[5] For example, when a node is only connected with two neighbours, the belief will have two distinct modes.

[6] Using UWB P400 RCM transceivers from Time Domain. Detailed information can be found at http://www.timedomain.com/p400.php.

[7] Two for the mean and three for the covariance matrix of the belief.

	SPAWN	SLEEP	ASLEEP
Message size	$4R_1$ bytes	20 bytes	20 bytes
Average number of messages per user per iteration	1	10.2	1
Average number of iterations before convergence	4	12	12
Average number of data packets per user	24	122.4	11
Complexity	$\mathcal{O}(R_1^2)$	$\mathcal{O}(R_2)$	$\mathcal{O}(R_2)$
Calculation time per user per iteration (on a 2.6 GHz processor)	7 sec	0.1 sec	0.1 sec

Table I
COMPARISON OF SPAWN, SLEEP AND ASLEEP

data, with ASLEEP, it becomes possible to just add some localization data to the header of a data packet. In an ideal case where every node in the network is actively transmitting data packets this would result in zero overhead for localization, i.e., no extra localization packets need to be transmitted. Note that this approach is less obvious for SLEEP because an individual message needs to be transmitted to each neighbour in a sequential order. To obtain the average total amount of packets sent by a user, the number of packets to represent a belief is multiplied with the number of iterations needed for convergence. In Table I the data load for all algorithms is compared for a LOS scenario. It can be observed that SLEEP requires the most packets to be sent over the network, i.e., on average 122.4 packets are sent per user. The altered SLEEP algorithm needs fewer packets, i.e., less than half the amount of messages compared to SPAWN and less than ten times fewer packets compared to SLEEP are needed. In order to asses the complexity of all algorithms, simulations were run in MATLAB on a 2.6 GHz processor that was fully dedicated to the algorithms. Table I shows that SPAWN takes about 70 times longer with the specified parameters than SLEEP and ASLEEP. From this we may conclude that, even on dedicated hardware with optimized code, SPAWN is probably unable to perform real-time localization.

V. CONCLUSIONS AND FUTURE WORK

We have investigated two message passing algorithms for cooperative localization: SPAWN and (an alteration of) SLEEP. The existing SLEEP algorithm is modified to reduce the network traffic. This altered version of SLEEP, called ASLEEP is compared with SPAWN for static localization under mixed LOS and NLOS conditions. In terms of accuracy, SPAWN performs best. However, the difference with ASLEEP is small. In terms of complexity, ASLEEP performs best with a complexity of order $\mathcal{O}(R)$ as compared to a complexity $\mathcal{O}(R^2)$ for SPAWN. When comparing the network data load for both algorithms, i.e., the number of packets transmitted over the network, the outcome will very much depend on the maximum packet size of the communication equipment. With currently available hardware, SPAWN requires twice as many packets to be sent over the network as compared to ASLEEP. Furthermore, with ASLEEP, it becomes possible to make a small modification to the header of a data packet to enable localization. With this it becomes possible to even further reduce localization-specific network traffic for some networks. Future research will include the comparison with some message passing algorithms, and a detailed overview of the network traffic and localization delay using a medium access protocol and a number of different censoring techniques to reduce network traffic [16].

REFERENCES

[1] S. Gezici, G. Giannakis, H. Kobayashi, A. Molisch, H. Poor, and Z. Sahinoglu, “Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks,” *IEEE Signal Processing Magazine*, vol. 22, pp. 70–84, Jul 2005.

[2] N. Patwari, J. Ash, S. Kyperountas, A. Hero III, R. Moses, and N. Correal, “Locating the nodes: cooperative localization in wireless sensor networks,” *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54–69, 2005.

[3] M. Rabbat and R. Nowak, “Distributed optimization in sensor networks,” in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pp. 20–27, ACM, 2004.

[4] H. Wymeersch, J. Lien, and M. Z. Win, “Cooperative localization in wireless networks,” *Proceedings of the IEEE*, vol. 97, pp. 427–450, Feb. 2009.

[5] M. Welling and J. J. Lim, “A distributed message passing algorithm for sensor localization,” *Proceedings of the 17th international conference on Artificial neural networks. ser. ICANN'07.*, pp. 767–775, 2007.

[6] T. Minka, “Expectation propagation for approximate Bayesian inference,” in *Uncertainty in Artificial Intelligence*, vol. 17, pp. 362–369, 2001.

[7] J. Lee and R. Scholtz, “Ranging in a dense multipath environment using an UWB radio link,” *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 9, pp. 1677–1683, 2002.

[8] H. Wymeersch, S. Marano, W. M. Gifford, and M. Z. Win, “A machine learning approach to ranging error mitigation for UWB Localization,” *IEEE Trans. Commun.*, 2012 (to appear).

[9] S. Gezici, H. Kobayashi, and H. Poor, “Nonparametric nonline-of-sight identification,” in *Vehicular Technology Conference, 2003. VTC 2003-Fall.*, vol. 4, pp. 2544 – 2548, Oct. 2003.

[10] H.-A. Loeliger, “An introduction to factor graphs,” *IEEE Signal Processing Magazine*, january 2004.

[11] J. Yedidia, W. Freeman, and Y. Weiss, “Constructing free-energy approximations and generalized belief propagation algorithms,” *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2282–2312, 2005.

[12] T. Minka, “Divergence measures and message passing,” *Microsoft Research Cambridge, Tech. Rep. MSR-TR-2005-173*, 2005.

[13] A. Ihler, J. Fisher III, R. Moses, and A. Willsky, “Nonparametric belief propagation for self-localization of sensor networks,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 809–819, 2005.

[14] T. Heskes, O. Zoeter, A. Darwiche, and N. Friedman, “Expectation propagation for approximate inference,” in *Proceedings UAI-2002*, pp. 216–233, 2002.

[15] T. Minka, *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[16] L. Chen, M. Wainwright, M. Cetin, and A. Willsky, “Data association based on optimization in graphical models with application to sensor networks,” *Mathematical and computer modelling*, vol. 43, no. 9, pp. 1114–1135, 2006.