Firefly-Inspired Synchronization for Energy-Efficient Distance Estimation in Mobile Ad-hoc Networks

Sabrina Merkel
Institute AIFB
Karlsruhe Institute of Technology (KIT)
76128 Karlsruhe, Germany
sabrina.merkel@kit.edu

Christian Werner Becker
Institute AIFB
Karlsruhe Institute of Technology (KIT)
76128 Karlsruhe, Germany
christian.becker3@student.kit.edu

Hartmut Schmeck
Institute AIFB
Karlsruhe Institute of Technology (KIT)
76128 Karlsruhe, Germany
hartmut.schmeck@kit.edu

***Abstract*—Mobile ad hoc networks (MANETs) are gaining increasing significance with computing devices becoming ubiquitous and equipped with wireless communication modules. Many applications for such networks require the devices to know their position within the network or their distance to other devices. Precise determination of these parameters often fails due to lack of information, missing hardware, or inaccessibility of needed resources, making an approximation necessary. We introduce an algorithm to calculate hop counts and, thereby, derive distances between devices. The algorithm is based on synchronization of all devices in the MANET. We show that an intentional phase shift of a periodically sent signal allows to estimate the distance between all devices in a network and a specific reference device. This approach significantly reduces the communication overhead leading to a more resource-efficient operation of the communication module and, thus, potentially extending the lifetime of the mobile devices. Experiments demonstrate that a network with an average of ten devices within communication range can be synchronized using a firefly-inspired decentralized synchronization algorithm. Also, we show that the resulting distance estimates have a higher accuracy compared to the results of an algorithm which is based on asynchronous exchange of messages.**

***Index Terms*—ad hoc networks; distance estimation; firefly algorithm; synchronization; nature-inspired computing; resource-saving; energy-efficient; communication;**

I. Introduction

In many applications such as geographic monitoring, smart buildings, target tracking, or disaster management, a large number of possibly mobile devices is utilized to accomplish a specific goal. In general, such devices consist of low power processors, have little memory and limited wireless communication range to exchange short messages with other devices. A network of such devices is called mobile ad hoc network (MANET) because the network's connectivity is dynamic and formed ad hoc. If a message in a MANET is sent to a device which is not situated within the communication range, the other network devices can be used as relays to forward the message [1]. An important parameter in many algorithms for MANETs is the hop count. The hop count denotes the number of relays two devices need at minimum to forward their messages to each other [2].

The hop count is mainly used in the context of routing [2]–[4], but it is also a basic parameter for many localization algorithms, because it can be used to estimate distances between devices [5], [6]. Localization is an important challenge for MANETs as the devices usually are not equipped with GPS, due to cost or size considerations. Applications which need localization are, for example, event reporting in a monitoring sensor network [7]–[9], location dependent routing [4], [10]–[14], assistance of group querying [15], pattern formation [16]–[19], and many more.

In this paper, we implement a nature-inspired synchronization method for devices of a MANET and propose an algorithm which determines hop counts using an intentional phase-shift of a periodically sent signal. Compared to a traditional asynchronous message-based hop counting algorithm, the accuracy of the derived distance estimates is shown to be higher. Furthermore, the synchrony of the devices enables energy-saving strategies for communication which makes the hop counting more resource-efficient. Besides improving the distance estimate through intentional timing of messages, the presented algorithm does not rely on complex message exchange and is, therefore, also suitable for devices with a very basic communication ability.

This paper is structured as follows: In Section II, the basics and the problem are described, as well as the related work concerning synchronization and hop counting in ad hoc networks. In Section III, the firefly-inspired hop counting (*FIHC*) approach is presented. Section IV shows the results of two experiments for testing the synchronization algorithm and for estimating distances. Section V concludes the paper.

II. Basics

A device's system clock is usually equipped with an oscillator [20], which creates almost harmonic oscillations with period T and, therewith, determines the frequency of the clock. The current state of an oscillating system can be described by its phase φ and we have $d\varphi/dt = 1/T$. When standardizing the phase φ to values between 0 and 1, the phase denotes the

percentage of the period which has elapsed. Thus, the phase can be expressed in terms of time t as depicted in Figure 1.

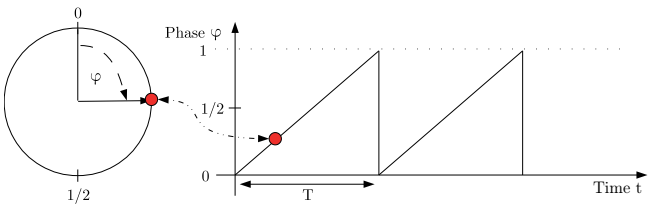


Fig. 1. Relation between a standardized phase φ and time t.

Timers are denoted as identical, if their period T is identical. Devices are denoted to be synchronous, if they have identical timers and their timers' periods are starting at the same time, i.e. their phase φ is identical for any point in time. Devices in MANETs are usually assumed to have asynchronous, identical timers. To achieve synchrony of these timers, we rely on a nature-inspired synchronization algorithm called firefly algorithm. In nature, synchrony can be observed in many different circumstances, e.g. heart cells are contracted in synchrony determining the natural heart beat and pulse [21], crickets chirp in synchrony [22], even the menstruation cycle of females adapt to each other striving for synchrony [23]. One of the most fascinating phenomenon can be observed in south-east Asia, where swarms of fireflies gather near trees and adapt the rhythm at which they emit light signals until they reach almost perfect synchrony [24]. The firefly algorithm imitates their behavior. The principle can be described as follows: Each oscillator is assumed to emit a signal at the end of its period, i.e. when its phase has the value 1. This process is also called firing. When another oscillator receives the firing signal, the time to its own next firing is shortened depending on the value of its phase at the time of reception. This might cause the oscillator to fire too. Figure 2 illustrates this mechanism which is explained in more detail below. Due to the described interaction, the oscillators are also called *pulse-coupled oscillators* [25].

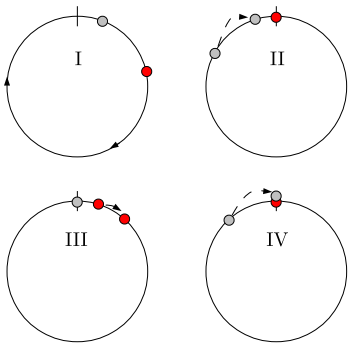


Fig. 2. Synchronization of two oscillators. The circles represent the identical period of the oscillators and the red and grey dots represent the phase of the two devices. The arrows indicate the phase-advance caused by the other device's firing.

The basic firefly algorithm was defined in [25] and has its origin in the work of Charles S. Peskin [21]. Each oscillator is described by a state variable $x \in [0, 1]$ which is often called *voltage* in later works, due to the intuition behind it. It is defined by an increasing, strictly concave function $f : \varphi \to x$ such that $\varphi = 0 \Leftrightarrow x = 0$, $\varphi = 1 \Leftrightarrow x = 1$ and $f' > 0, f'' < 0$. When the state x reaches the value 1, the oscillator fires and x is reset to 0. The reason for introducing the state variable is that it offers an easy way to adjust the sensitivity of the phase adjustment depending on the current phase. For example, when the state x is advanced by a fixed amount ϵ such that $x' = x + \epsilon$, then the phase is set as $\varphi' = f^{-1}(x')$ to reflect the new state of the device. In [25] the update function of the oscillators' states is defined as follows:

$$x_i = 1 \Rightarrow x_j \to \min(1, x_j + \epsilon), \forall j \neq i \tag{1}$$

Hence, when an oscillator i fires, the state of all other oscillators is increased by ϵ (cf. Equation 1), but has at most the value 1. We denote ϵ as the impulse-intensity. In [25] it is shown that any two oscillators can be synchronized with this simple rule. Also, for the case of $N > 2$ oscillators, it is shown that the amount of initializations which do not result in synchrony has a the Lebesgue measure 0. For this, it is not important, whether the impulse-intensity ϵ is considered as additive or not, when a group of oscillators fires at the same time. The only requirement is that the impulse-intensity is non-negative and is not zero for all oscillators simultaneously. In [26] it was shown that the system also converges, when the state-function f is non differentiable. In [27] the concavity of the state-function was shown to be an unnecessary restriction. It is shown that a linear state-function is sufficient for nearly all systems to converge if multiple synchronous firing is accounted for in an additive impulse intensity ϵ, and $\epsilon \geq 1/n$ holds.

Lucarelli and Wang show in [28] that a system also converges if there is only a neighborhood-coupling. This means that the firing of one oscillator can only be received by oscillators which are situated within a certain Euclidean distance around the emitting device. They use a linear state-function $f(\varphi) = \varphi$ and a different update model which can be described as follows:

$$x_i = 1 \Rightarrow x_j \to \min(1, x_j + \epsilon x_j), \forall j \neq i \tag{2}$$

With this update model the state advance is dependent on the current state of the affected device. If the state is close to zero, then this update rule will change the phase relatively little, while towards the end of the cycle the node will become more and more sensitive to incoming flash messages. Note, that with this state-function f the state x and phase φ are the same. Therefore Equation 2 can also be written as:

$$\varphi_i = 1 \Rightarrow \varphi_j \to \min(1, \varphi_j + \epsilon\varphi_j), \forall j \neq i \tag{3}$$

Experimental results show, that networks with such a linear model can also be synchronized if the network dynamically changes over time and even when it temporarily looses connectivity. This model closely relates to the MANET scenario,

which we selected for our hop counting algorithm. In [29] and [30] two algorithms are presented that also take delays into account. Nevertheless, for simplicity we base our hop counting algorithm on the synchronization according to the model of Lucarelli and Wang, neglecting message delays at first.

Nature-inspired synchronization has already been used successfully to implement an energy-efficient communication mechanism in MANETs [31]–[33]. Here, we use the synchronization as a precondition for a signal-based hop counting algorithm. The objective is that all devices in the network can derive their hop count, with respect to a reference device, from the phase shift of a second signal. The hop count refers to the number of relay nodes needed at minimum for two devices to communicate with each other. For sufficiently dense networks, this indicator also holds information about the distance of two nodes [5]. Many methods have been proposed to extract a distance estimate between nodes from known hop count and to use it for localization of the nodes [6], [34]–[43]. In the work presented here, we show that by a deliberately introduced asynchrony in an otherwise synchronous network, the hop count information can be encoded in the timing of an emitted signal. Furthermore, we develop a method to derive a distance estimate from the point of time of receiving a signal from other devices, and we show that this method can be used to save energy in the process of distance estimation. Additionally, it is shown that the resulting distance estimates have a higher accuracy compared to estimates produced via asynchronous message exchange.

III. Firefly-inspired Hop Counting (FIHC)

In distributed computing systems the exchange of messages is a major factor influencing the performance of the system and the lifetime of the devices involved [44], [45]. To reduce the energy consumption during hop count determination in a MANET, we present an algorithm called Firefly-Inspired Hop Counting (*FIHC*). In this section, we present the idea of the *FIHC*, show how the algorithm works, and discuss its advantages compared to an approach based on an asynchronous exchange of messages.

A. Concept

In literature, hop counts are calculated using an algorithm sometimes referred to as Gradient Algorithm [5], [38]. In the Gradient Algorithm, a node, called anchor node, starts by sending a message with value 0. The anchor node denotes the device to which all other devices are supposed to count their hops, i.e. to which they estimate their distance. When a device receives such gradient messages from its neighbors, it selects the minimum value it received, increments it by one and forwards it to its neighbors accordingly. The algorithm is repeated several times, before all devices in the network have calculated their hop count. In case of dynamic networks, the algorithm is repeated constantly, from time to time, to adapt to changes in the positions of the devices. The idea of firefly-inspired hop counting (*FIHC*) is to determine the hop counts in a network using a timed binary signal, instead of integer messages. This enables very simple devices to derive hop counts and, thus, their location within a network and reduces the message overhead, as well as the energy consumption of the devices. For *FIHC* we first need to synchronize all devices in the network. For this we use the firefly algorithm described in [28] because it describes a distributed way of synchronization and, therefore, is suitable for a MANET environment. When the network is in synchrony, a second signal is used which each device sends with a specific delay to the synchronized signal. The delay represents the hop count of the device, in a way that all devices sending the second signal at the same time also have the same hop count. Further on, we call the first signal, which is used for synchronization (*sync-signal*), and the delayed, second signal, which is used for hop counting (*hc-signal*). To be able to use signals for the transport of information we need to provide a method of encoding and decoding the information.

a) Encoding: To be able to encode the information about the hop count of a device in the phase shift, the phase shift has to be unique for any hop count. Assuming that a signal can be received immediately after sending, as long as the devices are situated within the communication range of each other, a distinct correlation between phase and hop count can be established as follows. If a device i is located h_i hops away from a reference device, its *hc-signal* is emitted, when its phase has the value $\frac{h_i}{h_i+1}$.

Lemma III.1. Any signal received at phase $\frac{h_i}{h_i+1}$ belongs to a device with distinct hop count h_i.

Proof: Let $h_i \geq 0, h_j \geq 0$ be the hop counts of device i and j, and $h_i \neq h_j$. It then holds:

$$\begin{aligned} & h_i \neq h_j \\ \Rightarrow\ & h_i * h_j + h_i \neq h_j * h_i + h_j \\ \Rightarrow\ & h_i * (h_j + 1) \neq h_j * (h_i + 1) \\ \Rightarrow\ & \frac{h_i}{h_i + 1} \neq \frac{h_j}{h_j + 1} \end{aligned}$$

Also, $\frac{h_i}{h_i+1} < 1$. ∎

For example, a device which is two hops away from a reference device emits its *hc-signal*, when the phase of the *sync-signal* has the value $\frac{2}{2+1} = \frac{2}{3}$.

b) Decoding: For decoding, Equation 4 is used. Let φ_i be the value of the *sync-signal's* phase of device $i+1$ when it receives the first *hc-signal* from its neighbors. First signal means, that it is the first signal received by any neighbor during the current period of the device's *sync-signal*. The hop count can then be calculated as:

$$h_{i+1} = h_i + 1 = \frac{-\varphi_i}{\varphi_i - 1} + 1 \tag{4}$$

for $0 \leq \varphi_i < 1$. The device would emit its own *hc-signal* when its *sync-signal* phase takes on the value $\varphi_{i+1} = \frac{h_{i+1}}{h_{i+1}+1}$.

The *FIHC* algorithm for mobile ad hoc networks now consists of two steps:

- Step I: Synchronization of all devices using the firefly algorithm
- Step II: Determination of the hop count through intentional phase shifting of the *hc-signal*.

For the first step we use the linear model of Lucarelli and Wang [28], because it allows for synchronization without the need of all-to-all coupling. Also, the algorithm is suitable for synchronization of temporarily disconnected networks and dynamically changing links between nodes, as experiments presented in [28] promise. The state-function f is linear and identical for all devices with $f(\varphi) = \varphi$, the phase φ and state x are identical and we will, therefore, directly refer to the phase change from here on. Each device sends a *sync-signal* (fires) at the end of its period, e.g. when its phase has the value 1. Figure 3 shows this procedure and the relation of phase and period.

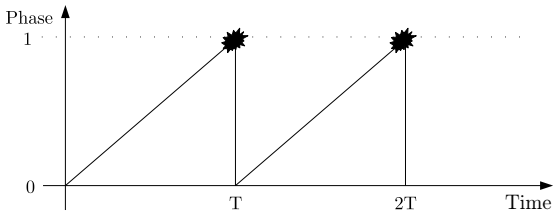


Fig. 3. Relation between phase and time. Devices fire when their phase has the value 1.

The interaction between a firing device j and a neighboring device i is shown in Equation 3.

Thus, the influence of a received firing signal on the phase of a device is not constant but depends on the current phase at the moment of reception. If a firing signal is received at the beginning of a period, the time until the receiving device fires next is shortened by a small amount. A firing signal received at the end of the period has a bigger influence and, under specific circumstances, leads to synchronization of the two devices. We assume that devices, where the phase is shifted to the value 1 due to the effect of another firing device, do not emit a signal at this moment but earliest at the end of the next period. In our model devices cannot distinguish a single firing device from a group of synchronously firing devices. We chose this constraint because in real applications it might technically be difficult to recognize how many signals are sent at the same time. Also, we assume that devices do not respond with phase shifts to received signals while they are firing, keeping the phase at value 1.

The determination of hop counts starts, when all devices are synchronous. It was not the main focus of this work to find decentralized mechanisms to determine whether a network is synchronous. Nevertheless, we run some experiments which indicate that, in many cases, it is sufficient when each device checks whether its neighborhood is in synchrony before starting the hop count process. For hop counting a second signal, the *hc-signal*, is used. It can be reasonable to use only one signal after successful synchronization if the network is a closed system, where no new devices join the network and have to be synchronized with the other devices.

We assume that all devices in the network want to determine their hop count with respect to one specific reference device. To do this, the reference device periodically sends the *hc-signal*, when the phase of its *sync-signal* has the value 0, e.g. at the beginning of its sync-period. A device i, which is not the reference device, has to emit its *hc-signal* when the phase of its *sync-signal* takes on the value of $\frac{h_i}{h_i+1}$, with h_i being the device's hop count. To find out the value of h_i the device listens to the earliest *hc-signal* that it receives from its neighbors and applies Equation 4. If $0 \leq h_i < h_j$, we have $\frac{h_i}{h_i+1} < \frac{h_j}{h_j+1}$ and, thus, the phase shift is smaller with lower hop count of a device.

Figure 4 shows an example of four devices which have to find their hop count, assuming that they are already synchronized. The reference device starts sending an *hc-signal* when the *sync-signal* phase has the value $\varphi_{ref} = 0$. When device I receives this signal its *sync-signal* phase is also $\varphi_I = 0$, so it knows that its hop count is 1 and emits its *hc-signal* at sync-phase $\varphi_I = \frac{1}{2}$. Device II receives the signal at sync-phase $\varphi_{II} = \frac{1}{2}$ and uses Equation 4 to calculate its own hop count with 2 and emits its signal at $\varphi_{II} = \frac{2}{3}$. Analogously, the fourth device III will send its hc-signal at sync-phase value $\varphi_{III} = \frac{3}{4}$.

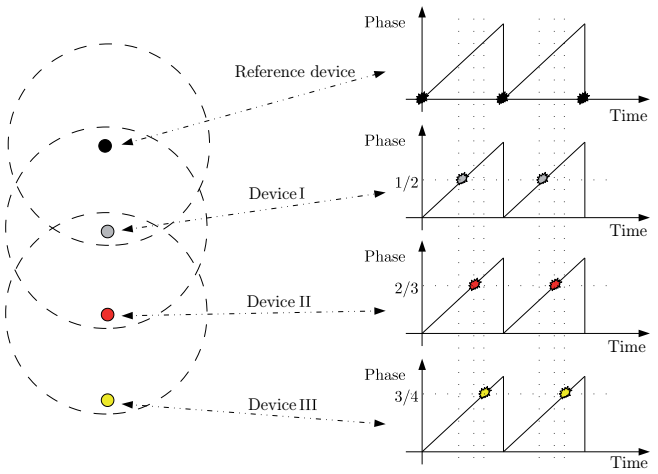


Fig. 4. Hop counting via intentional phase shifting.

The previously explained relation between phase value and hop count entails some significant benefits compared to hop counting via asynchronous message exchange:

1) Because of $h_i > h_j \geq 0$, it also holds that $\frac{h_i}{h_i+1} > \frac{h_j}{h_j+1}$ and $\frac{h_i}{h_i+1} < 1$. The devices, thus, emit signals in their ascending order of hop counts. Under the restriction that these signals are received by all devices within communication range without any delay and not considering network mobility, each device knows its hop count within one period of the *sync-signal*. The asynchronous message exchange algorithm, on the other hand, can take many periods (depending on the network size) until all

devices in the network determined their hop count.

2) Each device knows its hop count immediately after the first reception of a *hc-signal* from any of its neighbors and can ignore all other signals for the rest of its period. During this time the communication module is not needed for listening anymore and can be switched off to save energy. Asynchronous message exchange algorithms require all devices to constantly listen to new messages sent by neighbors.
3) Only the time of the signal reception is important and not the content of the message. Therefore, the message can be a binary value, as opposed to integer values needed for hop counting based on asynchronous message exchange. This reduces the length of the message, possibly reducing the use of the communication module even further, and it enables the hop count algorithm to be used for devices which have a very rudimentary communication module.

IV. Experimental Study

We performed two sets of experiments to analyze our proposed firefly-inspired hop counting method. The first set of experiments contributes some insights about the success of synchronization using the model proposed in [28] with neighborhood-coupling. The second set of experiments compares the signal-based hop counting with message-based hop counting, when used for distance estimation. We investigate the algorithm under static as well as dynamic conditions.

A. Scenario and Settings

For the experiments we consider a simulation of n identical devices placed randomly on a square plane according to a uniform distribution. The reference device is placed at the center of the environment and is supposed to be static during all experiments. Figure 5 shows the experiment scenario.

To handle the simultaneous mutual influences of the synchronization model, we decided to implement a time-discrete model. For this, the period is divided into m time slots. In each time slot, first, the phase of all devices is incremented by one time step (t = $\frac{1}{m}$), then, it is checked for all devices whether the phase of the device reached the value 1. If so, the device fires and all devices within communication range are marked. When the second step is finished, the phase of all marked devices is incremented according to Equation 3. We do not need to consider the influence of these devices on their neighborhood, because we assumed devices, which were shifted to a firing state, do not fire in the same period, but earliest one period later.

For the experiments, the period T is divided into $m = 100$ time slots. Devices can communicate, e.g. receive signals or exchange messages, when their Euclidean distance d is less or equal than a specific value r, which denotes the communication range. r is measured relative to the width of the environment, i.e. a range of 0.1 corresponds to 10% of the environment's width. This setting corresponds to the unit-disc model, widely used in ad hoc network simulation as a simplified model for the communication structure of a network. We do not take into account message delay, i.e. the signals or messages are assumed to be received at the moment of sending. To evaluate the performance of *FIHC* we implemented a very simple mobility model where each device moves with a probability of 1% in each time slot to a randomly chosen position which lies within a square of dimension $r \times r$ around its current position (cf. Figure 6). If the device would leave the environment due to a movement, the move is either only executed in the dimension, where the device does not leave the plane, or, if this is not possible, the device is not moved at all.

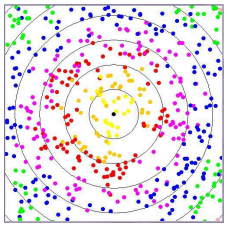


Fig. 5. Experiment scenario

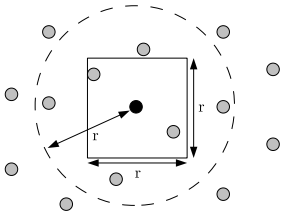


Fig. 6. Mobility pattern

B. Synchronization of Mobile Devices with Neighborhood-Coupling

In this section, we investigate the described synchronization algorithm in terms of the duration and success of complete network synchronization. To see the effect of the network size n (total number of devices), neighborhood density d (average number of neighbors per device), and impulse intensity ϵ, we test the following parameters:

n : 500 and 1000
d : 10, 20, 30, ..., 240 and 250
ϵ : 0.05, 0.10 and 0.15

The average number of neighbors d denotes the average amount of neighbors which a device has, that is located further than a distance of r from the border of the environment. We test all possible parameter combinations in both static and dynamic networks. Each experiment is repeated 30 times and an experiment is classified as failed if there is no complete synchronization after 300 periods. If and when the synchronization is successful, is determined in a centralized way by comparing the phases of all devices. Although, in practice, there should be a decentralized way for determining whether a network is synchronized, this is not subject to our investigations.

Figure 7 shows the percentage of successfully synchronized experiments in a static network of $n = 500$ devices with respect to different average neighborhood sizes d. As expected, the synchronization success increases with increasing impulse intensity ϵ until it reaches almost 100% for $\epsilon = 0.15$ throughout all regarded neighborhood densities. The neighborhood density d does not seem to have an obvious effect on the success rate. It slightly increases proportionally, especially for $\epsilon = 0.1$, but the results are very volatile and a clear

relation cannot be observed. In contrast to the neighborhood density the impulse intensity obviously has a strong impact. While the success rate fluctuates between 30% and 90% for $\epsilon = 0.5$, it is quite stable for $\epsilon = 0.1$, and is even more stable for $\epsilon = 0.15$. Figure 8 shows the time needed for synchronization for the successful experiments. The duration of synchronization decreases with increasing neighborhood density for $\epsilon = 0.05$ and $\epsilon = 0.1$. The degression can be explained easily, because the firing influences more devices when the neigbhorhood density is higher. On the other hand, this does not improve the success rate, so we conclude that neighborhood density accelerates synchronization but does not make it more probable. For $\epsilon = 0.15$, the increase of neighborhood density has first a positive impact and for approximately $d = 170$ turns into a negative impact. These two experiments show that a higher impulse intensity can increase the success rate for synchronization but also slows down the synchronization process for networks with high neighborhood density.

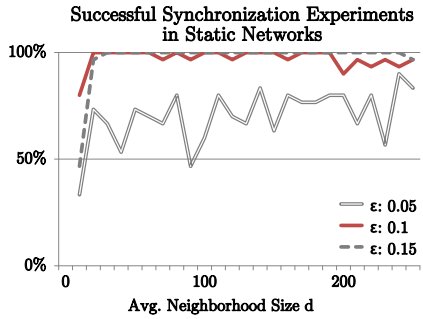


Fig. 7. Percentage of successfully synchronized experiments for static networks of size 500.

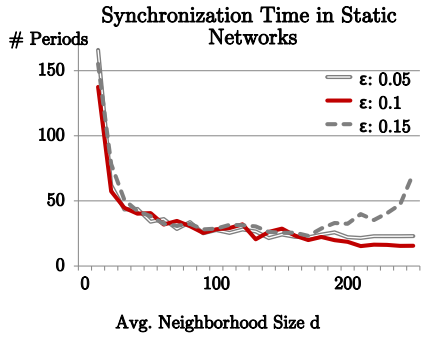
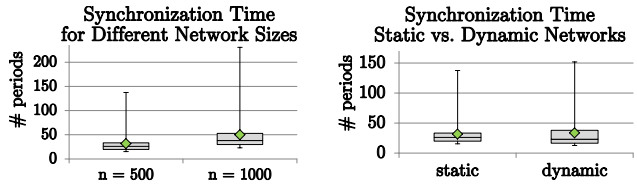


Fig. 8. Average synchronization time for static networks of size 500.

Because we reach a high success rate and simultaneously low synchronization time with $\epsilon = 0.1$, we use this impulse intensity for the remaining experiments.

The next experiment tests the influence of network size on the synchronization process. For this, we repeated the same experiment, as described before, in a network with 1000 nodes. The results for synchronization success differ only slightly with the trend that small neighborhood sizes in a larger network tend to have a lower success rate, whereas larger neighborhood sizes in larger networks also lead to slightly higher success rates. Overall, the mean average success rate for $n = 500$ lies at 97.3%, and for $n = 1000$ at 96.27%. Also the sample standard deviation is less for the smaller network size with 4.5% compared to 11.9% for 1000 devices. This indicates that a larger network is less likely to synchronize. This discovery meets the intuitive expectation that it is more difficult to synchronize a higher number of devices. When looking at the synchronization time, we see similar results. Figure 9(a) shows the results for both network sizes indicating that not only the success rate is worse in large networks, but also the duration for synchronizing the devices is higher. For further experiments we set the network size to 500 devices.

The final experiment shows the influence of mobility on the synchronization success and duration. The success of synchronization in mobile networks has a very similar behavior as the success rate for larger networks. The percentage of successfully synchronized experiments in dynamic networks is lower than in static networks when considering small neighborhood sizes. With higher neighborhood sizes the results change and dynamic networks synchronize more often than static ones. On average, both experiments show a success rate of 97.3% in static and 97.7% in dynamic networks. The sample standard deviation is 4.51% for static and 7.5% for dynamic networks confirming the similarity of dynamic and large-scale networks. The explanation is straight-forward. When a device in a network moves, the effects on the other devices are similar as if a new device would be added at the new position. In [46] the same effect is used to improve distance estimation results. Although the device was removed from its old position, it did influence those devices before and might already have led them to synchrony. Figure 9(b) shows the results for synchronization time, which is also higher for dynamic networks.



(a) Network size (b) Mobility

Fig. 9. Time for synchronization in networks of scale $n = 500$ and $n = 1000$ devices (a) and in static versus dynamic networks (b).

Overall, the experiments show that there is no guarantee for synchronization in the network with the model proposed. Further investigation indicate that this can be fixed when relaxing the constraint that multiple firing signals are received as one. The original model by Lucarelli and Wang takes additive signals into account and further investigations show that a network with an average neighborhood density of more than 9 devices and additive firing, always synchronizes. Also, the time for synchronization can be significantly reduced to 23 periods on average for static networks of size 500.

C. Distance estimation with firefly-inspired hop counting

The aim of the next experiment was to evaluate the performance of the *FIHC* algorithm while used in an ad hoc network application. For the application, we choose to use *FIHC* for distance estimation between all nodes in the network and one reference device, because this is a crucial part of many localization algorithms [5]. In literature, many hop count based distance estimation approaches have been proposed [6], [34]–[43]. Most of them are based on the idea to estimate the distance between a device i and j as:

$$\overline{d_{ij}} = h_{ij} * r \tag{5}$$

where h_{ij} denotes the hop count from device i to j. Although many improvements to this simple assessment have been published, we will use the simple estimation method to compare *FIHC* with an asynchronous message-based hop counting algorithm, because the distance is simple to compute and the deviation between the two methods becomes just as visible as with any refined estimation technique.

For the experiments we divide the *FIHC* method into two subcategories: *FIHC* and *FIHC delayed* (*FIHCd*). The difference is that in *FIHC* each device emits a signal according to its own hop count, whether or not it already received a signal in this period before. In case there is a signal after its own, the hop count is adapted, and the device emits a second *hc-signal* in the same period. As during *FIHC* all devices only react to the first received signal in a period, there can be a maximum of two *hc-signals* in the same period. The *FIHCd* algorithm does not send any *hc-signal* unless it receives one from another device (except for the reference device which always fires at phase 0). Thus, for the delayed method there will be only one *hc-signal* in each period. The third algorithm is the message-based version, where each device has an asynchronous timer and collects the messages from its neighbors during its timer's period. At the end of its period the device then calculates its hop count and sends it to all its neighbors.

For the *FIHC* and *FIHCd* experiment the timers are assumed to be synchronized. Hence, we do not take into account the synchronization process during the experiment. The experiments are performed in dynamic networks applying the mobility model described at the beginning of this section.

For evaluation the mean absolute percentage error (MAPE) was computed as $\frac{1}{n-1}\sum_{i\neq 0}\frac{|d_i-\overline{d_i}|}{d_i}$, with d_i being the Euclidean distance between the reference device and device i and $\overline{d_i}$ the estimate of this distance. The measurement starts, when all devices have updated their hop count 6 times to avoid initialization error, because it takes some time for the information to propagate from the reference device throughout the network, when asynchronous message-based hop counting is used. Again, experiments are repeated 30 times and the results are averaged.

Figure 10 shows the result for a network of size $n = 500$ and an average neighborhood size of $d = 20$ and $d = 100$ devices. In general, the MAPE is higher for the larger neighborhood size. This result seems unexpected at first, because a larger neighborhood usually provides more information and, thus, is expected to improve the result, but, because in our experiment setting a larger neighborhood also means a higher radius, the hop count error gets multiplied. For $d = 20$ the *FIHC* algorithm shows the lowest MAPE, followed by the message-based hop counting and the worst error is achieved using *FIHCd*. For $d = 100$ the order remains the same, although, the spread between the results is much less pronounced. The difference between *FIHCd* and the asynchronous message-based hop counting almost disappeared.

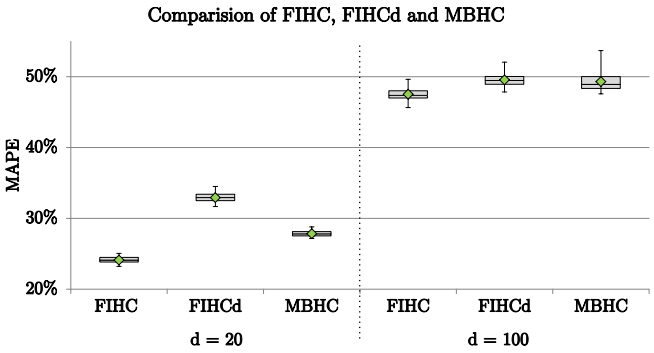
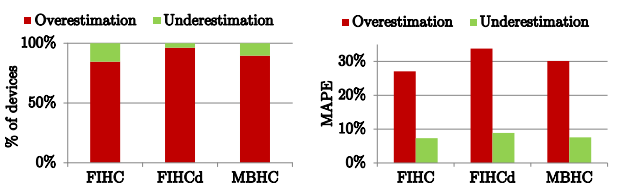


Fig. 10. Mean absolute percentage error (MAPE) of the three distance estimation algorithms for different neighborhood sizes.

To get an insight about the reasons for the performance deviations, we distinguish between over- and underestimate distances. Figure 11(a) shows the percentage of devices which over- and underestimated their distance respectively and Figure 11(b) displays the corresponding MAPE for a neighborhood size of $d = 20$. Three things become apparent. First, the percentage of overestimation is significantly higher than the percentage of underestimation for all three algorithms. Second, the MAPE is lower for underestimated distances, and, third, the performance order of the algorithms is reflected in the distribution of underestimations. The fact that more than 80% of the devices overestimate their distance, independently of the algorithm, is a result of the simple distance estimation method we applied. Due to the fact that the neighbor closest to the referece device usually does not lie at the border of the communication range, but somewhere closer to the device itself, the distances are naturally overestimated throughout the network (cf. [5], [47]).



(a) Percentage of devices which over- and underestimate the distance.

(b) Average MAPE of devices which over- and underestimate the distance.

Fig. 11.

Underestimation, due to the proposed distance estimation method, can only happen in a dynamic network. If a device

moves into a direction away from the reference device before updating its own hop count to the new situation, it can influence the surrounding devices in a way that they assume to have a smaller hop count than they would have in a static network. Figure 12 illustrates this effect. Here, three devices have a lower hop count than they would get in a static network. One, because it moved (orange star), the other two because they adapt their own hop count to the moved device (green stars). If a device, however, moves into the direction of the reference device, this has no influence on the surrounding devices because the hop count is always determined on the basis of the minimum hop count in the neighborhood (cf. Figure 13). As the distances are naturally overestimated, the mobility reduces the error for the distance estimates and, thus, the algorithms with higher proportion of underestimation have a lower MAPE. This effect is discussed in great detail for different kinds of mobility in [47].

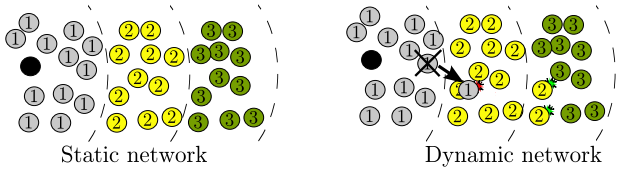


Fig. 12. The influence of movements away from the reference device on the hop count.

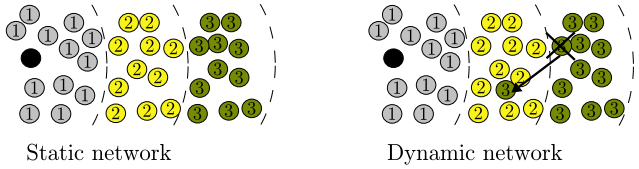


Fig. 13. The influence of movements towards the reference device on the hop count.

The effect of mobility is higher, the further away a device moves from the reference device before updating its hop count. The underestimation, thus, increases proportionally to the time between the hop count update and the signal emittance. For the *FIHCd* the maximum time between the hop count update and signal emittance is half a period:

Lemma IV.1. The time between receiving a firing signal φ_r and emitting a firing signal φ_e is at maximum $\frac{1}{2}$.

Proof: If device i receives its first signal in a period p, when its phase has the value $\varphi_r = \frac{k}{k+1}$, it updates its own hop count to the new value $k+1$ and emits a signal, when its phase takes on the value $\varphi_e = \frac{k+1}{k+2}$.

The time between the update and the signal emittance can be calculated as:

$$\frac{k+1}{k+2} - \frac{k}{k+1} = \frac{1}{(k+1)(k+2)} \overset{k\geq 0}{\leq} \frac{1}{2}$$

■

For *FIHC* the device also emits a signal when it did not receive one before, which is usually the case if the device has moved away from the reference node.

Lemma IV.2. If a device does not wait for the reception of a signal, the time difference between the hop count update φ_u and the signal emittance φ_e is more than one period.

Proof: If device i does not receive a signal from another device in period $p+1$ and it updated its hop count in period p to the value $k+1$, it then holds:

$$\frac{k+1}{k+2} + 1 - \frac{k}{k+1} = 1 + \frac{1}{(k+1)(k+2)} \overset{k\geq 0}{\geq} 1$$

■

This means that devices using *FIHC* by tendency moved further away from the reference device before emitting a new signal, compared to devices using the *FIHCd* algorithm (cf. Figure 14).

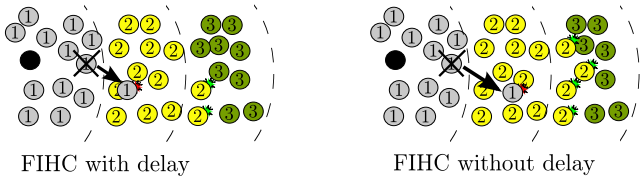


Fig. 14. Impact of moved distance on hop counts.

With the asynchronous message-based hop counting, the maximum time between the reception of the decisive message for the hop count and the signal emittance is at most one period. We suppose, that this is the reason, why the values for underestimation lie between the values of the other two algorithms.

Even though the experiments indicate that the *FIHC* algorithm is best used for distance estimation, this conclusion is misleading. *FIHC* does simply produce the highest amount of underestimation, which in turn corrects the natural overestimation, but we expect *FIHCd* to perform best, when the distance estimation method itself is more accurate, e.g. through any of the mentioned refinement methods. We expect this, because the hop counts are much more precise using *FIHCd*, due to fewer time between update and influence of other devices through signal emittance.

V. Summary and Outlook

We presented *FIHC* and its slight variation *FIHCd*, two algorithms for hop counting in mobile ad hoc networks, which are based on synchronized timers and simple signaling. The basic idea of *FIHC* is to use an intentional phase shift with respect to a synchronous base signal to encode information about the device's hop count with respect to a reference device. To realize the algorithm, we applied a nature-inspired synchronization method based on a firefly algorithm introduced by Lucarelli and Wang [28] and investigated the effects of different network parameters on the duration and success of synchronization. We found shortcomings, when devices were

not able to distinguish between the reception of a single and multiple synchronous signals. In this case, synchronization is still likely, but cannot be guaranteed. Another problem of *FIHC* is that, so far, the approach needs to use a different signal for each reference device. Also, the difference between two hop count signals becomes very small with increasing network size and, thereby, increasing hop counts. This can cause problems in distinguishing between successive times of reception. To avoid this problem, the period length can be increased, but this also increases the overall time for synchronization and hop count determination. Therefore, finding the right period length for the respective network is a challenge.
On the other hand, *FIHC* can accelerate the process of hop counting significantly compared to asynchronous message exchange based methods. It also reduces the use of the communication module, providing the possibility to save energy and extend the lifetime of the devices. Besides, the *FIHCd* version of the algorithm has been proven to have a shorter maximum time between update and signal emittance, which increases the accuracy of hop count based distance estimates, especially in dynamic networks. Even though the experiments indicate that the *FIHC* algorithm is best used for distance estimation, this is only due to the compensation of an overestimation-error induced by the simple distance estimation technique. When the distance estimation method itself is more accurate, the compensation is not necessary and *FIHCd* is expected to perform better in distance estimation, due to its lower degree of underestimation during mobility. The way *FIHC* was designed, it does not rely on the exchange of content between devices, but encodes information in the timing of signals, making it an alternative for localization of network devices with very basic communication abilities.
For future work, we want to explore methods for abandoning the need for employing a separate signal per reference device, e.g. through signal modulation techniques.

REFERENCES

[1] P. Ghosekar, G. Katkar, and P. Ghorpade, "Mobile ad hoc networking: Imperatives and challenges," *IJCA Special Issue on MANETs*, no. 3, pp. 153–158, 2010.

[2] P. Chatterjee and N. Das, "A distributed algorithm for load-balanced routing in multihop wireless sensor networks," in *Proceedings of the 9th International Conference on Distributed Computing and Networking*, ser. ICDCN'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 332–338.

[3] A. Boukerche, B. Turgut, N. Aydin, M. Z. Ahmad, L. Bölöni, and D. Turgut, "Routing protocols in ad hoc networks: A survey," *Computer Networks*, vol. 55, no. 13, pp. 3032–3080, 2011.

[4] C.-H. Chou, K.-F. Ssu, H. Jiau, W.-T. Wang, and C. Wang, "A dead-end free topology maintenance protocol for geographic forwarding in wireless sensor networks," *IEEE Transactions on Computers*, vol. 60, no. 11, pp. 1610–1621, 2011.

[5] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a global coordinate system from local information on an ad hoc sensor network," in *Information Processing in Sensor Networks*, F. Zhao and L. Guibas, Eds. Springer Verlag, 2003, vol. 2634, pp. 333–348.

[6] S. Y. Wong, J. G. Lim, S. V. Rao, and W. K. G. Seah, "Density-Aware Hop-Count localization (DHL) in wireless sensor networks with variable density," in *Proceedings of 2005 IEEE Wireless Communications and Networking Conference*, vol. 3. IEEE Computer Society, 2005, pp. 1848–1853.

[7] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 34–40, 2004.

[8] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *In Proceedings of the 7th symposium on Operating systems design and implementation*, ser. OSDI '06. USENIX Association, 2006, pp. 381–396.

[9] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou, "The challenges of building mobile underwater wireless networks for aquatic applications," *IEEE Network*, vol. 20, no. 3, pp. 12–18, 2006.

[10] C. Maihofer, "A survey of geocast routing protocols," *Communications Surveys Tutorials, IEEE*, vol. 6, no. 2, pp. 32–42, 2004.

[11] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *In Proceedings of the 6th annual international conference on Mobile computing and networking*, ser. MobiCom '00. ACM, 2000, pp. 120–130.

[12] K. N. Amouris, S. Papavassiliou, and M. Li, "A position-based multi-zone routing protocol for wide area mobile ad-hoc networks," in *In Proceedings of the 49th IEEE Conference on Vehicular Technology*, vol. 2. IEEE, 1999, pp. 1365–1369.

[13] J. C. Navas and T. Imielinski, "GeoCast - geographic addressing and routing," in *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, ser. MobiCom '97. ACM, 1997, pp. 66–76.

[14] W. Liao, Y. Tseng, and J. Sheu, "GRID: a fully Location-Aware routing protocol for mobile ad hoc networks," *Telecommunication Systems*, vol. 18, no. 1-3, pp. 37–60, 2001.

[15] J. Gehrke and S. Madden, "Query processing in sensor networks," *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 46–55, 2004.

[16] D. Coore, "Botanical computing: A developmental approach to generating interconnect topologies on an amorphous computer," Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, 1999.

[17] R. Nagpal, "Programmable self-assembly: Constructing global shape using biologically-inspired local interactions and origami mathematics," Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, 2001.

[18] W. J. Butera, "Programming a paintable computer," Thesis, Massachusetts Institute of Technology, 2002.

[19] M. Mamei and F. Zambonelli, "Programming pervasive and mobile computing applications with the TOTA middleware," in *Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications*. IEEE Computer Society, 2004, pp. 263–273.

[20] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: A survey," *Network, IEEE*, vol. 18, no. 4, pp. 45–50, 2004.

[21] C. S. Peskin, "Mathematical aspects of heart physiology," *Courant Institute of Mathematical Sciences, New York University*, pp. 268–278, 1975.

[22] T. J. Walker, "Acoustic synchrony: Two mechanisms in the snowy tree cricket," *Science*, vol. 166, no. 3907, pp. 891–894, 1969.

[23] M. J. Russell, G. M. Switz, and K. Thompson, "Olfactory influences on the human menstrual cycle," *Pharmacology, Biochemistry, and Behavior*, vol. 13, no. 5, pp. 737–738, 1980.

[24] J. B. Buck, "Synchronous rhythmic flashing of fireflies," *The Quarterly Review of Biology*, vol. 13, no. 3, pp. 301–314, 1938.

[25] R. E. Mirollo and S. H. Strogatz, "Synchronization of Pulse-Coupled biological oscillators," *SIAM Journal on Applied Mathematics*, vol. 50, pp. 1645–1662, 1990.

[26] R. Mathar and J. Mattfeldt, "Pulse-Coupled decentral synchronization," *SIAM Journal on Applied Mathematics*, vol. 56, no. 4, pp. 1094–1106, 1996.

[27] S. Bottani, "Pulse-coupled relaxation oscillators: From biological synchronization to self-organized criticality," *Physical Review Letters*, vol. 74, no. 21, pp. 4189–4192, 1995.

[28] D. Lucarelli and I. Wang, "Decentralized synchronization protocols with nearest neighbor communication," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '04. ACM, 2004, pp. 62–68.

[29] A. Tyrrell, G. Auer, and C. Bettstetter, "Fireflies as role models for synchronization in ad hoc networks," in *Proceedings of the 1st International Conference on Bio Inspired Models of Network, Information*

and Computing Systems, ser. BIONETICS '06. New York, NY, USA: ACM, 2006, pp. 1–7.

[30] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '05. New York, NY, USA: ACM Press, 2005, pp. 142–153.

[31] N. Wakamiya, S. Kashihara, and M. Murata, "A synchronization-based data gathering scheme in unstable radio environments," in *Networked Sensing Systems, 2007. INSS '07. Fourth International Conference on*, Jun. 2007, pp. 10–18.

[32] Y. Taniguchi, N. Wakamiya, and M. Murata, "A communication mechanism using traveling wave phenomena for wireless sensor networks," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2007, pp. 1–6.

[33] N. Wakamiya and M. Murata, "Synchronization-Based data gathering scheme for sensor networks," *IEICE Transactions on Communications*, vol. 88, no. 3, pp. 873–881, 2005.

[34] R. Nagpal, "Organizing a global coordinate system from local information on an amorphous computer," *MIT A.I. Laboratory*, no. A.I. Memo No. 1666, MIT, 1999.

[35] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proceedings of the IEEE Global Telecommunications Conference, 2001. GLOBECOM '01*, vol. 5. IEEE, 2001, pp. 2926–2931.

[36] ——, "DV based positioning in ad hoc networks," *Telecommunication Systems*, vol. 22, pp. 267–280, 2003.

[37] A. Savvides, H. Park, and M. B. Srivastava, "The n-hop multilateration primitive for node localization problems," *Mobile Networks and Applications*, vol. 8, no. 4, pp. 443–451, 2003.

[38] Q. Liu, A. Pruteanu, and S. Dulman, "GDE: a distributed gradient-based algorithm for distance estimation in large-scale networks," in *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '11. ACM Press, 2011, pp. 151–158.

[39] A. A. Ahmed, H. Shi, and Y. Shang, "A new hybrid wireless sensor network localization system," in *Proceedings of the 2006 ACS/IEEE International Conference on Pervasive Services*. IEEE Computer Society, 2006, pp. 251–254.

[40] P. Liu, X. Zhang, S. Tian, Z. Zhao, and P. Sun, "A novel virtual anchor node-based localization algorithm for wireless sensor networks," in *Proceedings of the 6th International Conference on Networking*. IEEE Computer Society, 2007, pp. 9–15.

[41] Q. Huang and S. Selvakennedy, "A range-free localization algorithm for wireless sensor networks," in *Proceedings of the IEEE 63rd Conference on Vehicular Technology*, ser. VTC '06, vol. 1. IEEE Computer Society, 2006, pp. 349–353.

[42] X. Bao, F. Bao, S. Zhang, and L. Liu, "An improved DV-Hop localization algorithm for wireless sensor networks," in *Proceedings of the 6th International Conference on Wireless Communications Networking and Mobile Computing*, ser. WiCOM '10. IEEE Computer Society, 2010, pp. 1–4.

[43] A. Karbasi and S. Oh, "Distributed sensor network localization from local connectivity: Performance analysis for the HOP-TERRAIN algorithm," *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, vol. 38, no. 1, pp. 61–70, Jun. 2010.

[44] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 5, no. 1, pp. 1–39, Feb. 2009.

[45] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[46] J. G. Lim and S. V. Rao, "Mobility-enhanced positioning in ad hoc networks," in *Proceedings of the 2003 Conference on IEEE Wireless Communications and Networking*, vol. 3. IEEE Computer Society, 2003, pp. 1832–1837.

[47] S. Merkel, S. Mostaghim, and H. Schmeck, "A study of mobility in ad hoc networks and its effects on a hop count based distance estimation," in *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, 2012, pp. 1–5.