An energy efficient joint localization and synchronization solution for wireless sensor networks using unmanned aerial vehicle

Wireless Netw (2015) 21:485–498
DOI 10.1007/s11276-014-0802-2

An energy efficient joint localization and synchronization solution for wireless sensor networks using unmanned aerial vehicle

Leandro A. Villas · Daniel L. Guidoni · Guilherme Maia · Richard W. Pazzi · Jó Ueyama · Antonio A. F. Loureiro

Published online: 4 September 2014

Abstract Localization and synchronization are fundamental services for many applications in wireless sensor networks (WSNs), since it is often required to know the sensor nodes' position and global time to relate a given event detection to a specific location and time. However, the localization and synchronization tasks are often performed after the sensor nodes' deployment on the sensor field. Since manual configuration of sensor nodes is usually an impractical activity, it is necessary to rely on specific algorithms to solve both localization and clock synchronization problems of sensor nodes. With this in mind, in this work we propose a joint solution for the problem of 3D localization and time synchronization in WSNs using an unmanned aerial vehicle (UAV). A UAV equipped with GPS flies over the sensor field broadcasting its geographical position. Therefore, sensor nodes are able to estimate their geographical position and global time without the need of equipping them with a GPS device. Through simulation experiments, we show that our proposed joint solution reduces time synchronization and localization errors as well as energy consumption when compared to solutions found in the literature.

Keywords Energy efficient · Wireless sensor networks · Unmanned aerial vehicle · Localization · Synchronization problems

L. A. Villas
Institute of Computing, University of Campinas, Campinas, Brazil
e-mail: leandro@ic.unicamp.br

D. L. Guidoni
Computer Science Department, Federal University of São João del-Rei, São João del Rei, Brazil
e-mail: guidoni@ufsj.edu.br

G. Maia · A. A. F. Loureiro
Computer Science Department, Federal University of Minas Gerais, Belo Horizonte, Brazil
e-mail: jgmm@dcc.ufmg.br

A. A. F. Loureiro
e-mail: loureiro@dcc.ufmg.br

R. W. Pazzi (✉)
Information Technology, University of Ontario Institute of Technology, Oshawa, Canada
e-mail: richard.pazzi@uoit.ca

J. Ueyama
Institute of Mathematics and Computer Science, University of São Paulo, São Paulo, Brazil
e-mail: joueyama@icmc.usp.br

1 Introduction

A wireless sensor network (WSN) can be defined as a cooperative network composed of thousands of small and resource-constrained sensor nodes [1]. These nodes are equipped with a wireless interface, processor, memory and sensing devices. Moreover, they have the ability to collect data about physical properties close to their physical location, such as temperature, humidity, pressure, movement and other properties. Despite the possibility for sensor nodes to harness energy from its surrounding environment [2–4], the attached battery still is the main power source. Therefore, solutions for this kind of network should focus on a low energy consumption footprint in order to maximize the network lifetime.

The main tasks of a WSN are the monitoring of physical phenomena and the transmission of the collected data to a special node, called sink node, through multi-hop communication. In this case, the sensor network is guided by events that generate data to be forwarded to the sink. Furthermore, in order to correlate events in space and time, it is necessary to have localization and synchronization systems in place.

Besides the task of localizing the collected data, some routing algorithms also use localization information to improve their performance, e.g., to reduce routing delays, number of hops, energy consumption and others, by creating routes that consider the node's position [5]. However, depending on the precision of localization information, such routes may not contain the correct nodes, thus decreasing the performance of these algorithms. Moreover, synchronization systems can also be used to increase the performance of routing protocols. There are some routing algorithms that consider a transmission delay/schedule to increase the routing performance [6]. Finally, some algorithms consider a joint localization and synchronization solution in their design. For instance, routing solutions that create overlapping routes to aggregate spatial-temporal correlated data [7].

Typically, a localization and synchronization solution uses a recursive approach [8–10]. In this scheme, a node estimates its localization and clock time, based on the position and clock time received from other nodes that already possess such information. Then, when a node is localized in space and time, it broadcasts its information to assist other nodes in their estimation. However, these solutions have some drawbacks. For instance, due to errors in the estimation process, after a node estimates its own position and clock based on the received and estimated information, it propagates the estimation error to other nodes. Furthermore, in a 3D scenario, a node must receive at least four positions from reference nodes to estimate its own position. Therefore, this may limit the number of nodes that are able to estimate their own position, since a node might not receive the proper amount of information to perform the estimation. Finally, to start the recursion process, 4–10 % of the network nodes must be equipped with a GPS receiver (beacon nodes) [8, 10]. This assumption increases the network cost, since it is reasonable to assume that the cost of a beacon node is much higher when compared to a node without a GPS receiver.

This work aims to eliminate some of the drawbacks described above, in particular, reducing the number of beacon nodes and avoiding the error propagation. For that, we propose a joint solution for a 3D localization and time synchronization system that uses an unmanned aerial vehicle (UAV) in WSNs. The UAV is equipped with a GPS receiver and traverses the sensor field broadcasting its position and clock time, allowing the sensor nodes to estimate their positions and clocks. The proposed solution exhibits three main contributions for localization and synchronization systems: (1) all network nodes are able to estimate their localization and local time with high accuracy, since the nodes receive information directly from the UAV; (2) the proposed solution is efficient for both sparse and dense networks, unlike most solutions in the literature that are affected by this network feature [10] and (3) the proposed solution reduces the network cost, since it is necessary for just one node (UAV) to be equipped with a GPS receiver.

2 Problem statement

Consider a WSN composed of n resource-constrained sensor nodes with a communication range r_c, which are scattered in a 3D field. Such network can be represented by an Euclidian graph $G = (V, E)$, with the following properties:

- $V = \{v_1, v_2, \ldots, v_n\}$ is the set of sensor nodes;
- $\langle i,j \rangle \in E$ if and only if v_i reaches v_j, i.e., the distance between v_i and v_j is smaller than r_c;
- $w(e)$ is the edge weight for edge $e = \langle i,j \rangle$, which corresponds to the distance between v_i and v_j.

Moreover, the following terms are used to describe the status of a sensor node during the localization and synchronization processes.

- *Unknown nodes—D:* the set of nodes that do not know their locations and their clocks are not synchronized;
- *Reference nodes—R:* the set of nodes that were able to estimate their positions and synchronize their clocks using the proposed solution. Therefore, the goal of any localization and synchronization algorithm is to turn *Unknown nodes* into *Reference nodes* by consuming the least amount of resources from the network;
- *Beacon nodes—B:* the set of nodes that do know their real physical positions and their clock is synchronized. This information is obtained through a GPS receiver or manual configuration. These nodes are the basis for most localization and synchronization systems for WSNs. Furthermore, they usually do not suffer from the same resource constraints as ordinary sensor nodes that need to be localized and synchronized.

Given the above terms, a definition for the joint localization and synchronization problem can be stated as follows:

Definition 1 **(Problem statement)** Assume a WSN represented by $G = (V, E)$. Furthermore, assume that for all $b \in B$, there is a set of *Beacon nodes B* with known positions (x_b, y_b, z_b) and synchronized clocks *(time_stamp_b)*. Therefore, the joint problem consists in finding $(x_u, y_u, z_u, clock_u)$ for the largest number of $u \in D$, thus converting *Unknown nodes* into *Reference nodes* by incurring a low communication overhead. Assuming that the communication is the most energy-intensive task in WSNs [11], the lower the communication cost (considering the number of transmitted and received messages) for the localization and synchronization algorithms, the lower the energy consumption of sensor nodes, thus increasing the network lifetime.

A straightforward solution for the aforementioned problem in a WSN is to equip all sensor nodes with a GPS receiver. Despite some clear advantages, such as relatively small localization and synchronization errors (2–15 m and 2–10 μs depending on the GPS receiver [5]) and high accuracy, since the errors would be similar for all sensor nodes in the network, this approach also possesses several drawbacks. For instance, the increase in the form factor of sensor nodes, the lack of visibility to satellites when used indoors, and finally and most serious, the increase in energy consumption and cost of sensor nodes. Therefore, this solution promptly becomes impractical for networks with hundreds or thousands of sensor nodes, which leads us to design and integrate a localization and synchronization system that consumes the least amount of resources from sensor nodes.

3 Related work

In this section, we describe some existing proposals in the literature. First, we present the literature's localization algorithms and then the synchronization solutions for WSNs.

3.1 Localization algorithms

Existing solutions in the literature, in one way or another, try to improve either the Ad Hoc position system (APS) [9] or the recursive position estimation (RPE) [8]. In the APS, a reduced number of beacon nodes (at least 3) is deployed in the sensor field. Each beacon node starts a broadcast message containing its position. Then, each unknown node calculates the distance from each beacon node using multi-hop communication. Once the distances are calculated, the unknown nodes can estimate their positions using, for instance, trilateration, thus becoming reference nodes. The RPE algorithm uses a different approach. The unknown nodes estimate their positions based on a set of beacon nodes (usually 5–10 % of the total number of sensor nodes). The algorithm is divided into four phases. In the first one, each beacon node sends its position to its neighbors. In the second phase, when an unknown node receives the beacon messages, it estimates the distance from each beacon using the RSSI technique. In the third phase, the unknown nodes estimate their positions based on the received information. Finally, in the fourth phase, the unknown nodes become reference nodes and send their positions to their neighbors; this increases the number of available positions by converting an unknown node into a reference node. The disadvantage of this algorithm is that the error in the position estimation is spread over the network; thus, the error increases during the estimation process performed during the localization process. There are other algorithms in the literature for the localization problem in WSNs. Most of them evolves from the APS and RPE algorithms by focusing on specific features of some scenarios [12–16]. For instance, [14] use a heterogeneous topology, where the beacon nodes have a powerful communication range. Galstyan et al. [15] use different techniques to compute a node position, such as a bounding box.

However, all the previously mentioned algorithms are interactive in nature, i.e., an unknown node must receive reference positions from beacons or reference nodes to calculate its position. This turns an unknown node into a reference node, and then it continues with this process to help other unknown nodes. Such fact increases the communication cost of these algorithms, thus leading to an increase in the energy consumption of sensor nodes, which might compromised the network lifetime. With this in mind, [17] proposed the LISTEN algorithm for localization of wireless camera sensor networks (WCSNs). In LISTEN, all unknown nodes need only to listen to the broadcast messages from a beacon node to determine their own locations. Such an approach decreases the communication cost of the localization task, thus decreasing the energy consumption. Our solution uses this same principle, i.e., unknown nodes need to just listen carefully to the broadcast messages from the UAV in order to calculate their positions. In our case, we use an UAV, which is a mobile element, to broadcast the location information to sensor nodes, whereas in LISTEN this information always comes from a fixed node. The advantage of our solution is that sensor nodes can improve their position information along the time, since they can receive multiple information packets. In case of LISTEN, the reception of multiple packets does not help improving the location information.

3.2 Synchronization algorithms

The problem of time synchronization can be divided into three cases: (1) relative time synchronization, which is used to order messages and events; (2) independent clock, where a node keeps track of drift and offset, and (3) global time synchronization, where there is a global time throughout the network. In this paper we are interested in the latter case. There are a number of synchronization algorithms available to solve global time synchronization in WSNs [18–23].

In [19], the authors propose the flooding time synchronization protocol (FTSP). The goal of FTSP is to synchronize the local clock of all network nodes based on a global clock. On such approach, initially, the clock of a given node is synchronized. Thereafter, this node transmits a message containing its timing information to its neighbors. After receiving such message, the neighbors are able to synchronize their clocks and the process continues until all nodes are synchronized. It is important to notice that, instead of using a fixed communication infrastructure, the node that starts the synchronization process broadcasts its synchronized clock to all other nodes in a multi-hop way using an ad-hoc structure. FTSP takes advantage of the MAC-layer time to send a message, called one hop synchronization (OHS). A root node, which has a synchronized global clock, creates a message with its clock and broadcasts this message to its neighbors. When an unsynchronized node receives this message, it gets the timestamp inside the message, then it adds a pre-defined OHS value to this timestamp and finally, it synchronizes its clock. Thereafter, the node broadcasts its synchronized clock to its neighbors. FTSP was evaluated in a real WSN and the OHS presented a precision of 2–4 μs in a Berkeley Mica2 platform [24].

In the FTSP protocol, the node transmits its time information after a predefined period of time. Such method is known in the literature as *slow-flooding*. In [21], the authors show that slow-flooding has some drawbacks regarding the accuracy and scalability. Therefore, they propose the flooding with clock speed agreement (FCSA) algorithm to overcome these drawbacks. For instance, FCSA forces all nodes to forward their time information using the same predefined period of time.

In [25], the authors propose the PulseSync algorithm, a new clock synchronization algorithm that is asymptotically optimal. The idea is to have a reference node to disseminate its clock as quickly as possible into the network to synchronize other nodes. [26] propose three schemes to achieve global clock synchronization: all-node-based, cluster-based and localized diffusion-based algorithms. In all three schemes, the number of exchanged messages among nodes is high due to several reference broadcast exchanges between a node and its neighbors. Thus, the schemes are not scalable when the number of nodes increases in the network. In [18], the authors propose an hierarchical approach to solve the time synchronization problem. In this hierarchical approach, there is a base station, the cluster heads and the ordinary nodes. The cluster heads and ordinary nodes do not have a synchronized clock. First, the cluster heads synchronize their clocks using the information from a base station. Thereafter, the ordinary nodes synchronize their clocks using the information provided by the cluster heads.

4 Proposed solution

This section presents the proposed solution to the 3D localization and synchronization problems using an UAV. Our solution aims to integrate both problems, since they are intimately related. Moreover, for a proper use of WSNs in real scenarios, it is necessary to solve both problems together. For instance, if a node has a synchronized clock but does not know its geographical location, it will probably not be useful for the network operation. In our solution, initially, all sensor nodes belong to set D and only the UAV belongs to the set B.

Figure 1 illustrates our proposed joint solution that uses an UAV. The UAV traverses the sensor field area where the nodes are deployed. During the flight, the UAV is responsible for periodically broadcasting its geographical position and its local timestamp. This information is based on the GPS receiver information.

After receiving four or more messages containing the position and timestamp from the UAV, the node is able to calculate its position and to synchronize its clock. Notice that, while in a 2D scenario it takes only three reference points to calculate the position of a node, in a 3D scenario, four reference points are required. Hereafter, we present all components of the localization and synchronization systems for the operation of our proposed solution. Furthermore, we show the integrated solution for a WSN using UAV.

4.1 Localization system

The localization system can be divided into two phases: distance estimation and position computation. First, we show how our solution estimates the distance between two nodes and then we show how it computes the nodes positions.

There are several methods to estimate the distance between two nodes in a WSN [5]. The most commonly used method is RSSI, since it requires no extra hardware besides a radio transmitter/receiver built into the sensor node. Unlike other distance estimation techniques [5], RSSI does not require any control message to estimate the distance between two nodes. That is, a node can estimate the distance between itself and another node based on the signal strength of any received data packet. Figure 2 illustrates the signal strength of a node when it sends a message considering three dimensions. The node sends out a signal with a certain power, which reduces as the signal is propagated. In this case, if the node that received the signal is further away from the node that sent it, then the received signal strength is low.

As already stated, the sensor node needs four reference points to calculate its position. These reference points are provided by the UAV during its flight over the sensor field.

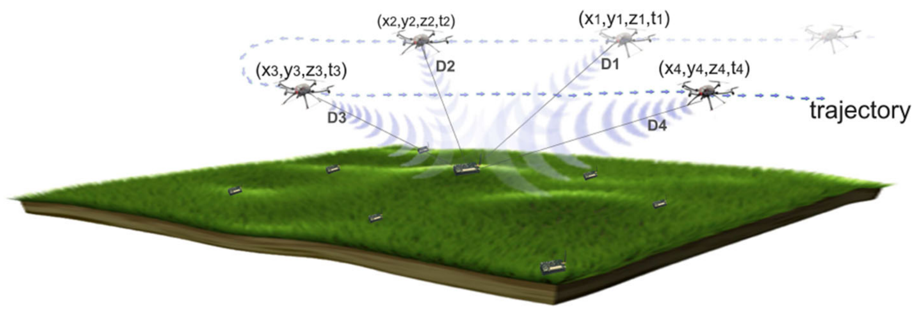


Fig. 1 Proposed solution

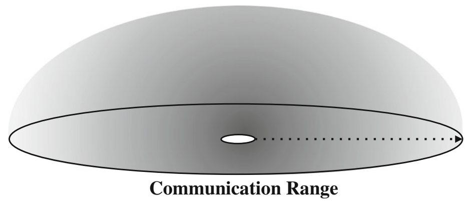


Fig. 2 Decrement in signal strength

Furthermore, the node also needs to know the distance between itself and each reference point. This can be performed by measuring the signal strength (RSSI) of each transmission from the UAV. Finally, when the sensor node has at least four reference points and has determined the distance to each point, it is able to estimate its position. Multilateration is the most common method used to estimate the position when a node is in possession of four or more reference points.

In the multilateration method, we have a system of at least four equations with three variables (x, y, z). Each equation in the system is constructed from the received reference positions and their respective distances. The system of equations can be represented as follows:

$$\begin{aligned}
(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 &= d_1^2 \\
(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 &= d_2^2 \\
&\vdots \\
(x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2 &= d_n^2
\end{aligned}$$

where (x_i, y_i, z_i) and d_i are, respectively, a reference position and the estimated distance to this reference position as calculated by the RSSI technique.

This system of equations is linearized by subtracting the last equation, that is $(x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2 = d_n^2$ from the other. Once linearized, we have a linear system that can be solved. In this work, the least squares [27] method was used to solve this linear system, since it is a simple and low cost method, which are important factors in designing solutions for WSNs. The solution of the linear system represents the position (x, y, z) of the node that performed the described steps.

4.2 Synchronization system

Some of the synchronization protocols are sender-receiver based and others are receiver-receiver based [28]. In the sender-receiver schema, the clock synchronization process is based on the time in which the message was sent [19]. On the other side, in the receiver-receiver schema, it is based on the time of arrival of the synchronization message [29]. In this work, we use a sender-receiver based protocol that reduces the number of messages to perform the clock synchronization of sensor nodes [28]. In a sender-receiver design, there are different ways to perform time synchronization. Some protocols use a two-way communication to discover the clock drift and, thus, correct the clock. However, as the goal of this work is to propose an integrated solution for both localization and synchronization problems, two-way communication is not applicable to our case. Instead, we use one-way communication, where the transmitter sends just one message and the receiver is able to synchronize its clock based on the time information contained inside the message.

For this purpose, we use the concept described in the FTSP, where the network nodes synchronize their clocks using one-way communication. To synchronize a clock with one-way communication, a node should calculate the following values: sender time, MAC access time, propagation time and receiver time. The sender time indicates the time to create a message to be transmitted into the network and the receiver time indicates the time required to receive a message and transmit it to the host. This time requirement can be softened if the timestamp is attached to the message in the MAC layer, just before its transmission. The propagation time can be easily calculated for a given propagation model. Finally, the MAC access time is the most difficult to calculate, since it depends on the network traffic and other network parameters. However, if the synchronization algorithm executes during the network startup, we may schedule the synchronization process without concurrent network tasks. We are able to do this because other tasks, such as routing protocols, depends on the synchronization process. In this case, the MAC access time is between 2 and 10 μs [19].

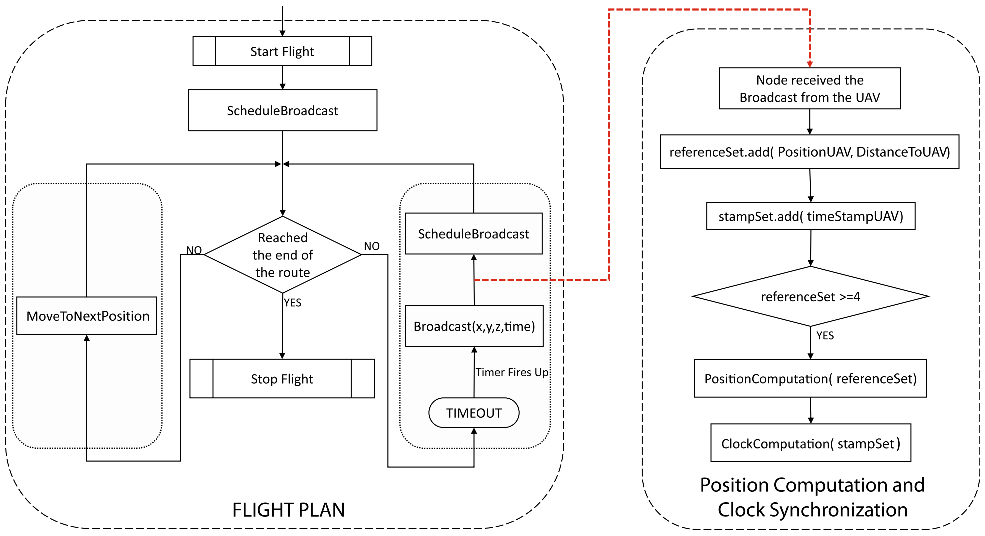

Start Flight
ScheduleBroadcast
Reached the end of the route
NO
NO
YES
MoveToNextPosition
Stop Flight
ScheduleBroadcast
Broadcast(x,y,z,time)
Timer Fires Up
TIMEOUT
FLIGHT PLAN
Node received the Broadcast from the UAV
referenceSet.add(PositionUAV, DistanceToUAV)
stampSet.add(timeStampUAV)
referenceSet >=4
YES
PositionComputation(referenceSet)
ClockComputation(stampSet)
Position Computation and Clock Synchronization

Fig. 3 A flow chart showing the main components of the proposed solution

4.3 Joint solution using UAV

As illustrated in Fig. 1, the operations of the proposed solution are divided into two phases. The first one refers to the UAV transversing the sensor field. The second phase is related to the position computation and clock synchronization. When a node receives a message from the UAV, it calculates its distance from the UAV using the RSSI technique and stores the position and timestamp of the UAV. When a node has at least four messages, it is able to calculate its 3D position and synchronize its clock. As a node receives four timestamps, it uses all received timestamps to decrease the estimation error in order to synchronize its clock. Figure 3 shows the main steps of the proposed solution. The first phase is described in the left dotted rectangle (flight plan) and the second phase (position computation and clock synchronization) is described in the right dotted rectangle.

A flight plan contains the airplane route, which is previously designed by the network designer. After takeoff, the algorithm schedules the transmission of the UAV's position and timestamp. While the end of the route is not reached, the algorithm retrieves the next point where the UAV should move, which in turn flies to the specific point with a certain speed. It is important to highlight that the UAV performs a periodic broadcast in parallel with its displacement over the sensor field.

When a node receives a message from the UAV, it calculates the distance to the UAV using the RSSI technique. Afterwards, the node retrieves the UAV position from the received message and stores the position and distance to the related position in *referenceSet*. The UAV timestamp is stored in *stampSet*. If the number of received positions is >4, the node is able to compute its position and synchronize its clock. To compute its position, the node uses the least squares method described above. To compute its local time, the node makes an average of all received timestamps. Also, for each received timestamp, the function adds a predefined OHS error, which is the error related to the MAC access time and propagation time.

It is important to point out that our proposed joint solution can be executed anytime during the network lifetime. However, in this paper, we assume that it is executed during the network startup. This is a reasonable decision, since other network tasks may require the nodes' positions and a synchronized clock, such as geographic routing. Notice, however, that due to the introduction of new nodes in the network or because of the nodes' clock drift (which may introduce an error in the clock of the node after a period of time), the algorithm can be re-executed during the network lifetime to estimate new positions or to (re)synchronize the nodes' clock.

5 Simulation results

5.1 Scenario description

The proposed 3D localization and synchronization integrated solution is compared to some solutions found in the literature; however, each one solves each problem individually. In order to do a better comparison, we combined a localization solution with a synchronization solution from the literature. We evaluated how easily it would be to put them together to solve the 3D localization and synchronization problems. We identified the recursive position estimation algorithm (RPE) and the FTSP as the most appropriate ones. The RPE is the basis for a number of position estimation algorithms found in the literature. The choice of the FTSP is based on the structureless communication approach used by the authors. Thus, both protocols can use the same message exchange procedure without the need of changing any of them. As described in Sect. 3, the FTSP uses one-way communication to synchronize the node clock. In this way, when a beacon or reference node sends its position in the RPE algorithm, we add the local timestamp of the node into the localization message. Based on this information, the node may estimate its position and synchronize its clock. It is important to point out that an unknown node becomes a reference node when it computes its position and clock. This can only be done after receiving four messages, since we are studying the 3D localization problem. In this case, the node will synchronize its clock only when it has enough information to compute its position. This combined strategy leads to the RPE–FTSP solution.

The main goal of our performance evaluation is to assess the proposed integrated algorithm considering the following metrics: (1) energy consumption, (2) estimation position error, (3) synchronization error (μs), and (4) number of nodes that were not capable of estimating their positions and synchronizing their clocks. To do this, we vary four important network parameters: (1) number of network nodes; (2) network density, (3) RSSI error, and (4) OHS error. We evaluated three different flight plans (see Fig. 4) named spiral, sinusoidal and linear paths. In the analysis, we noticed that the flight plan does not significantly affect the assessed metrics. The flight plan only affects the time required to cover the whole monitored area. In other words, it affects the required time for all nodes in the network to estimate their positions and to synchronize their clocks. Assuming it is of utmost importance that the sensor nodes estimate their positions and synchronize their clocks in the shortest possible time, therefore, to carry out the evaluations, we employed the sinusoidal flight plan, illustrated in Fig. 4(b), which had the shortest time to cover the whole monitored area. The simulation parameters are presented in Table 1. The communication range of the sensor node and the UAV is 50 m. This was set in order to have a fair comparison with the literature algorithms. The energy consumption needed to transmit a packet is 0.08 J (Joules) and the energy consumption needed to receive a packet is 0.02 J. The packet length is composed of the packet header and the payload and has a total of 568 bits. The packet header has 440 bits and the payload has 128 bits. The payload is divided into four values of 32 bits (a float value of four bytes), which correspond to the position (x, y, z) and the *timestamp*. All these values are based on the MicaZ node [30].

To simulate the RSSI technique, we measure the distance between the sender and receiver and we introduced an error, which varies from 0 to 20 % of the real distance between the two nodes. The number of beacon nodes in the RPE–FTSP integrated solution varies from 25 to 200. Beacon nodes are equipped with a GPS receiver. It is important to note that in our integrated solution, the UAV is the only node equipped with a GPS receiver. The monitoring area (x, y) is considered as the Eq. 1,

$$x = y = \sqrt{\frac{n \times pi \times r_c^2}{Density}} \tag{1}$$

where n is number of nodes, r_c is the communication radius, and *Density* is the average degree of neighbors. For each simulation in which the number of nodes varies, the monitoring area size is adjusted accordingly in order to maintain the node density at the same value. The third dimension (z) for each node is a random number between 0 and 10 m. The UAV altitude is a random number between 20 and 50 m, which changes with each new direction while traversing the monitoring area. We use the SinalGo v.0.75.3 [31] simulator to evaluate the algorithms. Each scenario was replicated 33 times with different seeds for the random number generation. In all results, the curves represent the mean values, whereas the error bars represent the confidence interval of 95 %.

5.2 Overall energy consumption

In this section, we evaluate the overall energy consumption for both RPE–FTSP and the proposed joint solution. To carry out this evaluation, we fixed the number of nodes to 500 and the network density to 30. The goal of this analysis is to verify the energy consumption of the nodes that execute the localization and synchronization algorithms to estimate their positions and synchronize their clocks. Therefore, in the RPE–FTSP integrated solution, we do not consider the energy consumption of the beacon nodes, since these nodes will spend more energy due to the GPS receiver. In this case, beacon nodes can be equipped with more energy reserves. In our solution, we also do not consider the energy consumption of the UAV.

Figure 5 shows the map of energy consumption for both algorithms considering a different number of beacon nodes in the RPE–FTSP algorithm. This energy map is obtained after executing the algorithms and considering that the beacon nodes are randomly deployed in the network. Figure 5(a) shows the energy map for our joint solution. We can verify that the energy consumption of the proposed solution is homogeneous for all network nodes. This result can be explained by the fact that the UAV broadcasts its position to all nodes while it transverses the monitoring area and a sensor node does not communicate with its neighbors to estimate its position and to synchronize its clock. Figure 5(b), (c), (d) show the energy map considering the RPE–FTSP for a different number of beacon

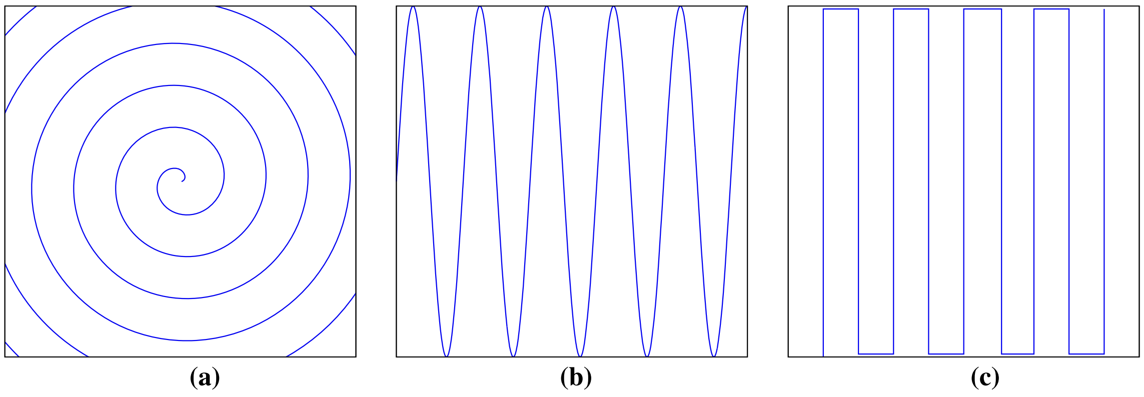


Fig. 4 Flight plans. **a** Spiral path, **b** sinusoidal path, **c** linear path

Table 1 Simulation parameters

Parameters	Values
Number of nodes	250–2,000
Density	15–50
Communication range	50 m
UAV communication range	50 m
UAV speed	10 m/s
RSSI error (%)	0–20
OHS error (μs)	0–30
RPE–FTSP	25–200 beacon nodes
Monitoring area (x and y)	$x = y = \sqrt{\frac{n \times \pi \times r_c^2}{Density}}$
Terrain (z)	0–10 m
Flight altitude	20–50 m
Broadcast msg interval (UAV)	1/s
Energy to transmit	0.08 J
Energy to receive	0.02 J
Packet length	568 bits

nodes. It is important to point out that the energy consumption of the RPE–FTSP is not homogeneous among the nodes, which may cause some nodes to run out of energy earlier than others. Moreover, when we increase the number of beacon nodes, the energy consumption also increases. This is due to the fact that the RPE–FTSP algorithm depends on (1) the communication among sensor nodes; (2) the number of beacon nodes, and (3) the positions of the beacon nodes in the monitoring area. When the number of beacon nodes is small, less messages are transmitted in the network, thus leading to a lower energy consumption. This fact is not observed when we increase the number of beacon nodes. When the network has more beacon nodes, an unknown node will receive more messages than the number required to estimate its position and synchronize its clock, thus, causing a great impact on the energy consumption. Notice that, even when the number of beacon nodes is small, if they are geographically close to one another, an unknown node will spend more energy by receiving all the broadcast messages from the beacons.

In the following sections, we analyze, among other metrics, the energy consumption, taking into consideration the impact of the number of nodes and network density.

5.3 Number of nodes

In this section, we evaluate the solutions for a different number of network nodes. For this analysis, we fixed the network density to 30, the RSSI error to 5 % and the OHS to 5 μs.

Figure 6(a) shows the position estimation error. We can see that, the proposed system has a small error and it is not affected by the number of nodes, which is not observed in the RPE–FTSP algorithm. The RPE–FTSP position estimation error is around three times greater when compared to our proposal and increases when we increase the number of nodes. This happens because when fixing the number of beacons and increasing the number of nodes, the unknown nodes estimate their position based on reference nodes, which have an estimated position. Thus, the estimation error spreads in the network. We also can observe that when we increase the number of beacon nodes, the estimation error decreases, since more unknown nodes will estimate their position using beacon positions. It is important to point out that, when we only have 25 beacon nodes, the RPE–FTSP algorithm is not able to estimate any position when $n > 500$. The main disadvantage of using many beacon nodes is the network cost, which increases substantially because of the GPS receivers. Also, when the localization and synchronization problems are solved, the beacon nodes become useless, since this process runs just once during the network lifetime.

The synchronization error is shown in Fig. 6(b). When we increase the number of network nodes, the synchronization error of the RPE–FTSP algorithm also increases.

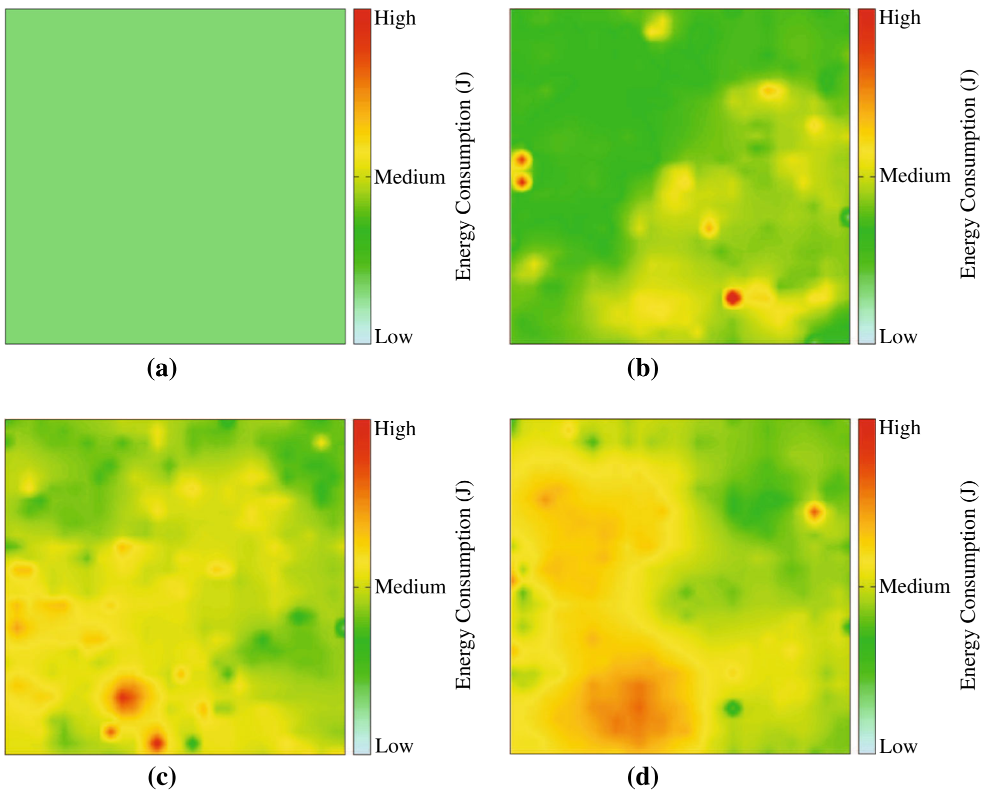


Fig. 5 Overall energy consumption. **a** Proposed solution, **b** RPE–FTSP with 50 beacon nodes, **c** RPE–FTSP with 100 beacon nodes, **d** RPE–FTSP with 150 beacon nodes

This is due to the same fact as discussed above. When the network has 2,000 nodes, the synchronization error of the RPE–FTSP algorithm is 1.89 times greater than that of our proposed solution (when $B = 200$). We also can see that the proposed synchronization system is not affected by the number of nodes.

Figure 6(c) shows the number of unknown nodes. A node is labeled as unknown when it does not receive enough information to estimate its position/clock or when its position error is greater than its communication range. We can observe that when the network has a few beacon nodes, the number of unknown nodes increases when the number of network nodes also increases. However, when the network has up to 750 nodes, the number of unknown nodes is low. It is important to point out that as the UAV traverses the entire monitoring area, all network nodes are able to estimate their positions and to synchronize their clocks.

The energy consumption is illustrated in Fig. 6(d). We can verify that when the network has between 250 and 500 nodes, the number of beacon nodes has a great impact on the energy consumption. As we described above, when the network has a small number of sensor nodes and a high number of beacon nodes, an unknown node receives more messages than the amount necessary to compute its position and clock. When the number of nodes is >750, the number of beacon nodes does not have an impact on the energy consumption, since the monitoring area is large enough to decrease the density of beacon nodes. For instance, when the number of beacons is >750, the energy consumption of the RPE–FTSP is more than 2.5 times greater when compared to the proposed solution.

5.4 Network density

This section evaluates the algorithms for different network densities. The number of network nodes is 1,000, the RSSI error is 5 % and OHS is 5 μs. Figure 7(a) shows the error in the position estimation process. We can observe that the higher the values on the network density, the better the RPE–FTSP performance. This is due to the fact that when we increase the network density for a fixed number of nodes, the monitoring area decreases. In this case, the position estimation error does not spread to many nodes. Our solution, which uses an UAV, is not affected by the network density, since the UAV transverses all the monitoring area.

The same behavior is observed in the synchronization problem, since both algorithms execute together [Fig. 7(b)]. It is important to notice that for higher values of network density, there is no difference between our approach and the RPE–FSTP algorithm. Figure 7(c) shows

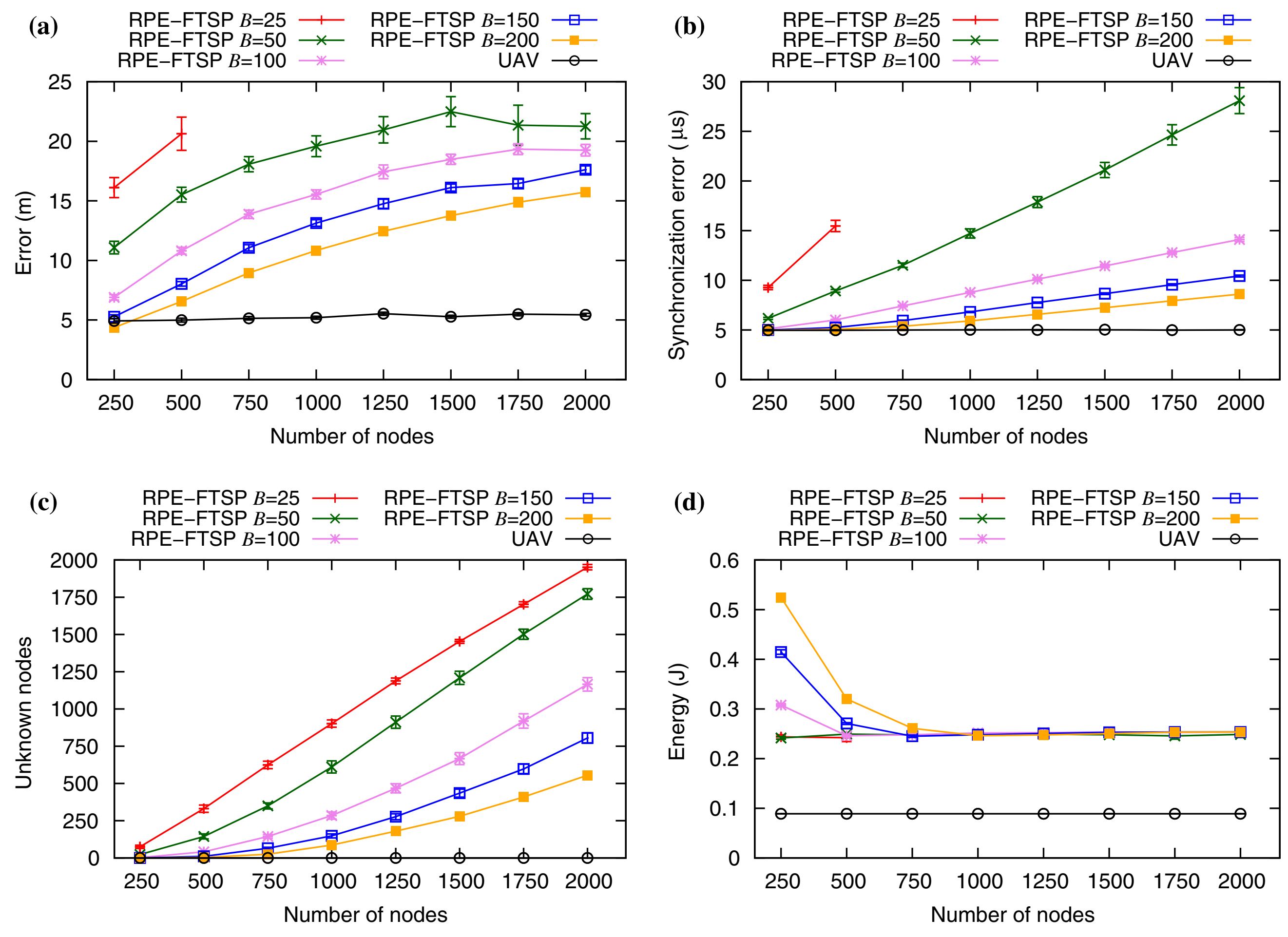


Fig. 6 Number of nodes. **a** Position error, **b** synchronization error, **c** unknown nodes, **d** energy consumption

the unknown nodes. As expected, when we increase the network density, the number of unknown nodes decreases quickly when the number of beacon nodes is >50. For a network with a high level of density and a great number of beacon nodes, the unknown nodes are <10 % of the nodes. Considering our proposal, the number of unknown nodes is zero, since the UAV transverses the entire monitoring area broadcasting its position and timestamp.

Figure 7(d) shows the energy consumption for different network densities. When the network density is up to 30, the energy consumption in the RPE–FTSP is not affected by the number of beacon nodes. For network densities >30, when we increase the number of beacon nodes, the energy consumption also increases. When the network density is 50, the energy consumption, considering 200 beacon nodes, is 1.27 times greater than with 25 beacon nodes. It is important to note that the proposed solution is not affected by the network density. Considering 200 beacon nodes, the error in the position estimation of the RPE–FTSP is just 1.5 times greater. When compared to our proposed solution, the synchronization error is the same, but the energy consumption is 3.75 times greater.

5.5 RSSI error

To better understand the behavior of the localization algorithms, we introduced an error in the RSSI technique, which varies from 0 to 20 % of the distance from the sender. The number of network nodes is 1,000 and the network density is 30. Figure 8 shows these results. We can note that the position estimation error is zero when the RSSI error is zero. However, this scenario is not realistic. For all values of RSSI error (except 20 %), our proposal overcome the RPE–FTSP algorithm. This is due to the fact that in the RPE–FTSP algorithm, the localization error spreads amongst the nodes. However, when the RSSI error is high (representing a very noisy environment), the results of both RPE–FTSP and UAV solutions are closer. When the RSSI error is 20 % and in the case of 200 beacon nodes, both solutions have the same results.

5.6 OHS error

In this section we analyze the impact of the OHS error in the synchronization problem (see Fig. 9). We used the

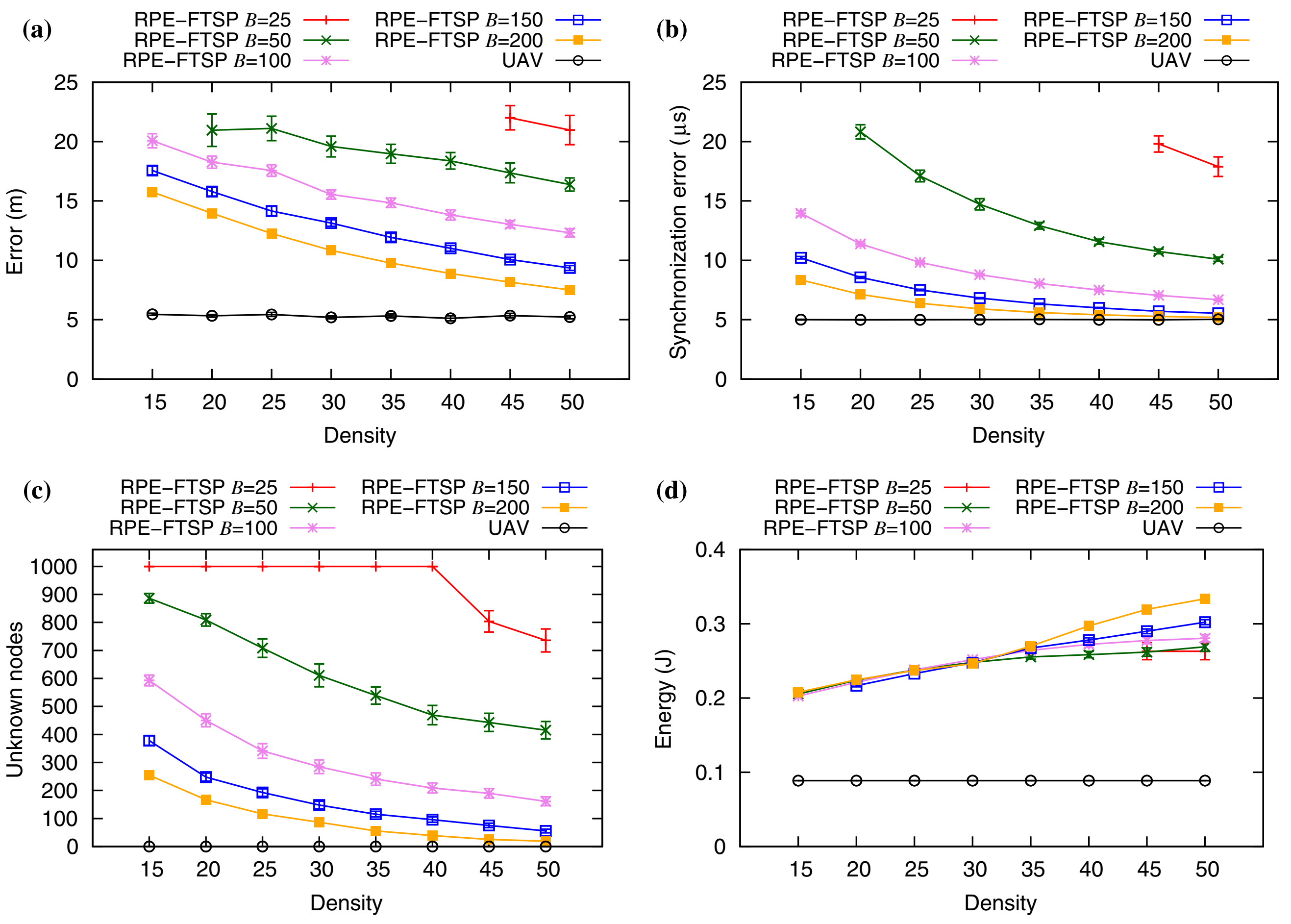


Fig. 7 Network density. **a** Position error, **b** synchronization error, **c** unknown nodes, **d** energy consumption

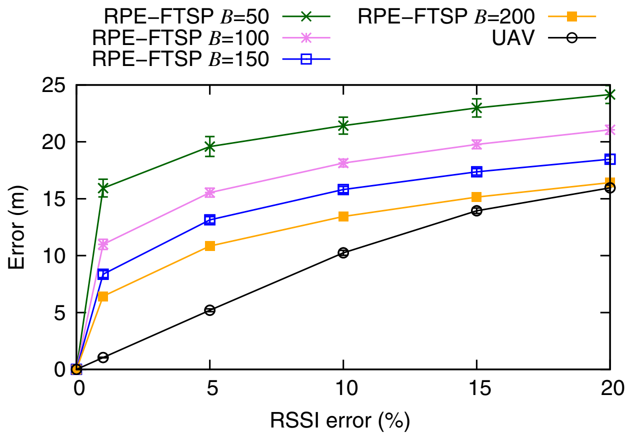


Fig. 8 RSSI error

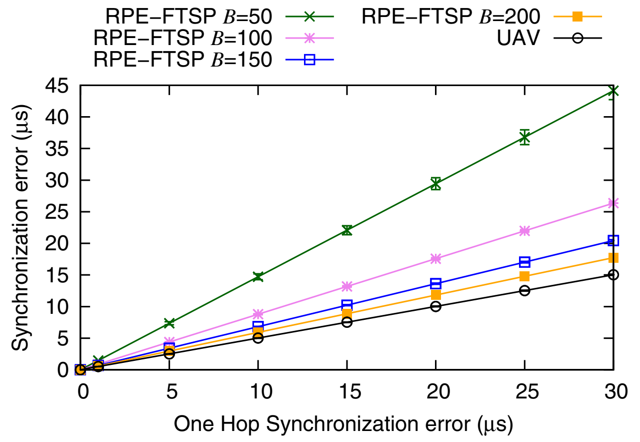


Fig. 9 OHS error

same scenario described in the previous section. As expected, when we increase the OHS error, both RPE–FTSP and our solution possess a greater synchronization error. As discussed above, the synchronization error spreads with the localization error in the RPE–FTSP algorithm, which does not happen in our proposal, since the UAV sends its information directly to the sensor nodes. It is important to highlight that our proposal presents better results when compared to any number of beacon nodes in the RPE–FTSP algorithm (except when $B = 200$).

6 Conclusions

Localization and synchronization are fundamental services for many WSN applications. There is a large class of applications that need to associate localization and time information to the sensed data. Typically, in the approaches found in the literature, the estimation error depends on the number of beacon nodes deployed on the sensor field. Moreover, these beacon nodes significantly increase the cost of the network. Furthermore, most solutions do not consider the problem of 3D localization. In this work, we proposed a joint solution for the 3D localization and synchronization problems in WSNs by using an UAV. The UAV traverses the sensor field broadcasting its geographical position and clock time, allowing the sensor nodes to estimate their positions and global time. Simulation results show that the proposed solution reduces synchronization and localization errors when compared to existing protocols. Moreover, the efficiency of our solution is independent of the number of nodes in the network, which is an important aspect in the case of scalability. Finally, in our approach, all sensor nodes are able to calculate the global time and estimate their positions.

As future work, we intend to propose a new flight plan to reduce the time required to cover the entire monitored area and to conduct experiments in a real environment.

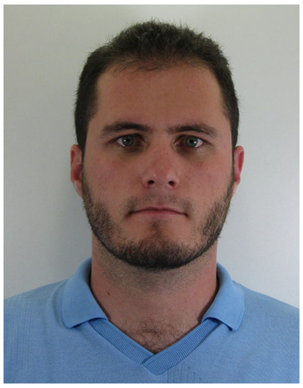
Acknowledgments This work is partially supported by FAPESP (processes 2012/22550-0 and 2013/05403-66), CNPq, CAPES, FAPEMIG (process APQ-01947-12), and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cyirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks*, *38*(4), 393–422.
2. Jiang, X., Polastre, J., & Culler, D. (2005). Perpetual environmentally powered sensor networks. In *IPSN '05* (pp. 1–10).
3. Zeng, K., Ren, K., Lou, W., & Moran, P. J. (2009). Energy aware efficient geographic routing in lossy wireless sensor networks with environmental energy supply. *Wireless Network*, *15*(1), 39–51.
4. Liu, R. S., Fan, K. W., Zheng, Z., & Sinha, P. (2011). Perpetual and fair data collection for environmental energy harvesting sensor networks. *IEEE/ACM Transactions on Networking*, *19*(4), 947–960.
5. Boukerche, A. (2008). *Algorithms and protocols for wireless sensor networks*. Hoboken: Wiley
6. Akkaya, K., & Younis, M. (2005). A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, *3*(3), 325–349.
7. Villas, L. A., Boukerche, A., Guidoni, D. L., de Oliveira, H. A., de Araujo, R. B., & Loureiro, A. A. (2013). An energy-aware spatio-temporal correlation mechanism to perform efficient data collection in wireless sensor networks. *Computer Communications*, *36*(9), 1054–1066.
8. Albowicz, J., Chen, A., & Zhang, L. (2001). Recursive position estimation in sensor networks. In *ICNP '01* (pp. 35–41).
9. Niculescu, D., & Nath, B. (2003). Ad hoc positioning systems (aps) using oao. In *IEEE INFOCOM '03* (pp. 1734–1743).
10. Oliveira, H. A. B. F., Boukerche, A., Nakamura, E. F., & Loureiro, A. A. F. (2009). Localization in time and space for wireless sensor networks: An efficient and lightweight algorithm. *Performance Evaluation*, *66*(3–5), 209–222.
11. Gupta, P., & Kumar, P. (2000). The capacity of wireless networks. *IEEE Transactions on Information Theory*, *46*(2), 388–404.
12. Wang, G., & Yang, K. (2011). A new approach to sensor node localization using rss measurements in wireless sensor networks. *IEEE Transactions on Wireless Communication*, *10*(5), 1389–1395.
13. Ma, D., Er, M. J., & Wang, B. (2010). Analysis of hop-count-based source-to-destination distance estimation in wireless sensor networks with applications in localization. *IEEE Transactions on Vehicular Technology*, *59*(6), 2998–3011.
14. Guidoni, D. L., Boukerche, A., Villas, L. A., Souza, F. S. H., de Oliveira, H. A. B. F., & Loureiro, A. A. F. (2012). A small world approach for scalable and resilient position estimation algorithms for wireless sensor networks. In *MOBIWAC '12* (pp. 71–78).
15. Galstyan, A., Krishnamachari, B., Lerman, K., & Pattem, S. (2009). Distributed online localization in sensor networks using a moving target. In *IPSN '09* (pp. 61–70).
16. Deak, G., Curran, K., & Condell, J. (2012). A survey of active and passive indoor localisation systems. *Computer Communications*, *35*(16), 1939–1954.
17. He, Y., Liu, Y., Shen, X., Mo, L., & Dai, G. (2013). Noninteractive localization of wireless camera sensors with mobile beacon. *IEEE Transactions on Mobile Computing*, *12*(2), 333–345.
18. Yao, H., & Zhou, W. (2010). Synchronization algorithm for multi-hop in wireless sensor networks. In *CIS '10* (pp. 28–32).
19. Maróti, M., Kusy, B., Simon, G., & Lédeczi, A. (2004). The flooding time synchronization protocol. In *SenSys '04* (pp. 39–49).
20. Liu, Y., Li, J., & Guizani, M. (2012) Lightweight secure global time synchronization for wireless sensor networks. In *IEEE WCNC '12* (pp. 2312–2317).
21. Sinan Yldrm, K., & Kantarc, A. (2014). Time synchronization based on slow flooding in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, *25*(1), 244–253.
22. Wu, Y. C., Chaudhari, Q., & Serpedin, E. (2011). Clock synchronization of wireless sensor networks. *IEEE Signal Processing Magazine*, *28*(1), 124–138.
23. Wu, J., Jiao, L., & Ding, R. (2012). Average time synchronization in wireless sensor networks by pairwise messages. *Computer Communications*, *35*(2), 221–233.
24. Mica2 crossbow technology. Mica2 DataSheet. Document Part Number: 6020-0042-0. Resource document. http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf. Accessed 2 Aug 2014.
25. Lenzen, C., Sommer, P., & Wattenhofer, R. (2009). Optimal clock synchronization in networks. In *SenSys '09* (pp. 225–238).
26. Li, Q., & Rus, D. (2006). Global clock synchronization in sensor networks. *IEEE Transactions Computers*, *55*(2), 214–226.
27. Golub, G. H., & Loan, C. F. V. (1996). *Matrix Computations* (3rd ed.). Baltimore, MD: Johns Hopkins University Press.

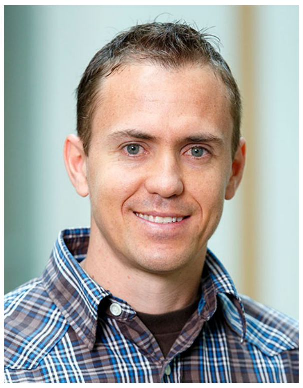
28. Sundararaman, B., Buy, U., & Kshemkalyani, A. D. (2005). Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, *3*(3), 281–323.
29. Elson, J., Girod, L., & Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. *SIGOPS Operating Systems Review*, *36*, 147–163.
30. Micaz crossbow technology. MicaZ DataSheet. Document Part Number: 6020-0060-04 Rev A. Resource document. http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf. Accessed 2 Aug 2014.
31. Distributed Computing Group, ETH Zurich. Sinalgo: Simulator for network algorithms. http://dcg.ethz.ch/projects/sinalgo. Accessed 2 Aug 2014.

Leandro A. Villas is an Assistant Professor at Institute of Computing of The University of Campinas, Campinas, Brazil. He received his Computer Science degree from the Colleges Integrated Adamantinenses in 2004, Adamantina, Brazil, the Master Degree in Computer Science from The Federal University of Sao Carlos in 2007, São Carlos, Brazil, and the Ph.D. in Computer Science from The Federal University of Minas Gerais in 2012, Belo Horizonte, Brazil. He received his Postdoctoral from The University of Sao Paulo in 2013. His research interests include context interpretation, distributed algorithms, routing algorithms, wireless sensor networks and vehicular ad hoc networks. He spent a year in the lab PARADISE SITE, University of Ottawa as part of his doctoral sandwich. Over the past 5 years, he had eight papers in journals, more than 30 papers at conferences; he received award for best article in the XXV Brazilian Symposium on Computer Networks and Distributed Systems 2007 and prominent researcher award in 2011 at PARADISE Lab, University of Ottawa.



Daniel L. Guidoni is an Assistant Professor at Federal University of São João del-Rei, Minas Gerais, Brazil. He received his Ph.D. in Computer Science from Federal University of Minas Gerais, Belo Horizonte, Brazil in 2011. In 2012, he received the CAPES award for the best Ph.D. thesis in Computer Science, Brazil. His research interest includes Wireless Sensor Network, Small World Networks, Vehicular Networks and communication protocols.

Guilherme Maia is a Postdoctoral Researcher in Computer Science at the Federal University of Minas Gerais, Brazil. He got his Ph.D. in Computer Science at this same university in 2013. His research interests include distributed algorithms, mobile computing, wireless sensor networks and vehicular ad hoc networks.



Richard W. Pazzi is an Assistant Professor at the Faculty of Business and Information Technology, University of Ontario Institute of Technology, Canada. Prior to joining UOIT, Dr. Pazzi was appointed as senior research manager of the NSERC DIVA Strategic Research Network, from 2010 to 2012, to oversee the network activities and conduct research in the area of intelligent vehicular networks. His research interests include fault-tolerant protocols for Wireless Sensor Networks and Mobile Computing. He is also active in the areas of Vehicular Ad Hoc Networks, multimedia communications and networked 3D virtual environments. He is the recipient of Best Research Paper Awards from the IEEE International Conference on Communications (ICC 2009) and the International Wireless Communications and Mobile Computing Conference (IWCMC 2009), and the recipient of Elsevier's Top Cited Article (2005–2010) for his work published in the Journal of Parallel and Distributed Computing (JPDC 2006).

Jó Ueyama completed his Ph.D. in computer science at the University of Lancaster (England) in 2006. He is currently an Associate Professor at the Institute of Mathematics and Computer Science, University of São Paulo (ICMC/USP); and he is also a Brazilian Research Council (CNPq) fellow. His main research interest includes Computer Networks and Distributed Systems.

Antonio A. F. Loureiro received his B.Sc. and M.Sc. degrees in computer science from the Federal University of Minas Gerais (UFMG), Brazil, and the Ph.D. degree in computer science from the University of British Columbia, Canada. Currently, he is a full professor of computer science at UFMG, where he leads the research group in wireless sensor networks. His main research areas are wireless sensor networks, mobile computing, and distributed algorithms. In the last 10 years he has published regularly in international conferences and journals related to those areas, and also presented tutorials at international conferences.