Where’s the Mote? Ask the MoteHunter!

Csaba Kiraly and Gian Pietro Picco

DISI – University of Trento, via Sommarive 14, 38123 Trento, Italy
{csaba.kiraly, gianpietro.picco}@disi.unitn.it

Abstract—Contrary to laboratory environments, real-world wireless sensor network deployments face harsh conditions where motes can be lost during deployment or in operation, for several reasons. Motes mounted on animals can easily detach. Fixed motes could get displaced by environmental conditions, e.g., heavy rains. These motes could contain valuable data and/or equipment, but finding them out in the wild could be quite challenging. Similar challenges arise in the cases where the placement of nodes is not known a priori, and yet in-field interaction with them (e.g., for data downloading or debugging) is needed.

We present MOTEHUNTER, a tool supporting in-field searching for motes, composed of: the Hunter, a special node with custom hardware and software and equipped with a directional antenna, and the Prey, which can be in principle any mote compliant with IEEE 802.15.4, although a special small-footprint software component can be integrated with the application to simplify the search. We illustrate the architecture of MOTEHUNTER and discuss our design choices quantitatively and qualitatively.

Index Terms—Wireless sensor networks, directional antennas, real-world deployments, localization, IEEE 802.15.4.

I. INTRODUCTION

“Where’s the mote?” The question is not unheard when wireless sensor networks (WSNs) are applied in-field.

A motivating example. In our recent experience, we encountered the problem of rescuing a mote lost in the field in a real-world deployment on animals. The ultimate goal of our project is long-term monitoring of the behavior of wildlife, specifically roe deer living in the mountains nearby Trento. To this end, we designed a custom hardware platform (built around a Telos-like WSN node) and the related software (TinyOS-based), along with the collars packaging the WSN nodes. However, as capturing roe deer is cumbersome and expensive, we decided to first test and evaluate our system on cows and horses, more easily accessible.

We expected hardware and software bugs to creep up during these preliminary tests, which indeed happened. However, we faced an additional, unexpected challenge: the collar. Indeed, a collar designed for long-term use must be designed carefully, to avoid hurting the animal. As a result, our “production” collars are expensive to build and, therefore, during the short-term tests on cows and horses we opted for a lower-quality variant, faster and cheaper to assemble. The downside of this decision, however, was that collars were easily worn out by the behavior of the animals, a few to the point of breaking the collar and detaching it (and the node) from the animal. Had the animals been living in a barn, this would not be too much of a problem. However, to test our system in realistic conditions, the animals we chose were allowed to move freely in a relatively wide range around the farm, which was located in an uninhabited area in the mountains. As a result a few nodes were lost. We retrieved some of them by sheer luck through “random” searching, but we actually lost a couple for good.

The situation will be exacerbated in our final deployment on roe deer, as their movements are unconstrained, and actually determining the (large) area where they dwell is one of the goals of the project. A GPS device will be installed onboard but activated only sparingly (to save battery), therefore yielding only a very approximate idea of the potential target area. Even if the GPS remains operational after it had detached from the animal, its signal could be too weak due to e.g., dense vegetation or wrong antenna position. Further, we also plan to apply the same architecture to smaller species, where GPS is not an option due to the added weight imposed by the batteries guaranteeing its operation.

Why search for a mote? Our primary motivation for MOTEHUNTER was to rescue a mote lost in the field. Other applications share the same concern as ours, e.g., in the cases where the nodes, albeit fixed, are weather-beaten, detach from their supports, get washed away, or otherwise move from their intended location. The cost of a lost node is an important factor, although this is not always the case in the grand scheme of things. For instance, our fellow biologists sometimes\(^1\) do not bother retrieving collars far more expensive than ours, as the cost of recapturing the animal is far higher than the cost of the collar. On the other hand, the true reason to rescue a mote is often the data it carries. Depending on the application, it may have stored important data (e.g., in our case, contacts with other animals) that cannot be reported otherwise. Or, if the node was malfunctioning, getting your hands on it may be the only hope to identify the culprit.

The need to search for a mote, however, is not motivated solely by its loss. For instance, in some applications the placement of nodes is not performed by an operator, and is random, as in scenarios where nodes are dropped from some kind of robot or aerial vehicle. Data download by an operator in the field must be supported by the ability to find the motes without a precise location. Even in the aforementioned wildlife application, if the nodes do not contain a GPS, direct interaction with the nodes (e.g., to download application or debugging data) is possible only if these are first “found”, i.e., if the operator is close enough. This in-field search for radio beacons has actually been used for decades by zoologists.

\(^1\)In some cases this is not possible due to concerns about polluting components, e.g., those in batteries and some electronics.
especially on tiny species like mice [1], which are tracked by means of a tiny (analog) transmitter attached to them, and a big VHF antenna held by an operator walking in the general area of interest.

**Contribution and roadmap.** In this paper we propose a simple solution to the aforementioned problem in the form of MOTEHUNTER, a hardware and software design and prototype for locating nodes in the field, which revolves around the use of a directional antenna. As evident from the description above, unlike the vast literature on localization [2], our goal is not to provide automated, accurate positioning, rather provide support (e.g., general direction) to an in-field human operator. Driven by our immediate needs, our reference WSN platforms are Telos or equivalent, equipped with a radio compliant with IEEE 802.15.4, and running TinyOS.

Our design is inspired by very simple and practical goals, which we illustrate in Section II. Section III concisely illustrates the core principles of the operation of MOTEHUNTER. The hardware and software architecture of the operator node, called the Hunter, is illustrated in Section IV and V, respectively. The searched node, called the Prey, can be any node whose radio is compliant with IEEE 802.15.4. Section VI presents the functionality that MOTEHUNTER, through its graphical user interface, makes directly available to the operator, in terms of both quantitative measurements and qualitative visual clues. Section VII discusses some quantitative insights about the relationship between the chosen directional antenna and the effectiveness of the mote-searching process. Section VIII describes opportunities for extending our tool with additional capabilities. Section IX places our work in the context of related ones. Finally, Section X ends the paper with concluding remarks.

**II. Design Goals**

MOTEHUNTER is composed of two elements. The main one is the so-called “Hunter”, the tool used to search for motes. It is the combination of special purpose hardware, a TinyOS application similar to a sniffer, and a Java graphical user interface.

Of course, there is no fun in hunting if there is no prey. The second element, which we call “Prey”, can be any IEEE 802.15.4 compliant mote. The Hunter works even if no modifications are made to this mote. It is enough to know that it is out there to start hunting it down. However, we did develop a small software-only TinyOS component that can optionally be embedded in the Prey’s code to facilitate the hunt.

We set for ourselves the following guidelines for the design of our Hunter tool:

- use of COTS equipment: although we could have started with a new hardware design with specialized equipment (using antenna arrays and signal processing techniques), we decided to build our solution from commercial off-the-shelf components. More specifically, we tried to create a cheap and readily available solution using components that are usually already on stock in research groups working on WSNs;
- easily usable in the field: we aim at the design of a tool that someone can easily bring along while doing field work. Therefore, it should be relatively compact;
- toolset approach: during the normal operation of a WSN, nodes do not get lost. It is in the exceptional cases that our tool should get handy. Unfortunately there could be many of these exceptions, largely different in their nature and in the solution required. Therefore, our tool should resemble more of a swiss army knife than a monolithic tool;
- intuitive GUI: We design a graphical user interface that integrates these search tools, presenting measurements in a variety of ways. The goal of the GUI is not the presentation and logging of exact measurements, but an intuitive visualization of the gathered data;
- backup option for extreme situations: as stated above, exceptions are our target. We should be prepared for unexpected situations in the field, where sometimes everything that could go wrong actually does. The notebook’s battery might be down, the directional antenna might be left in the car, or the compass might break. We should provide backup options that make the tool useful even in these challenging conditions.

**III. Principles of Operation**

We locate the Prey by taking advantage of the directionality of the Hunter’s antenna, by recording RSSI measurements, and by using the acknowledgement feature of IEEE 802.15.4. As the Hunter’s antenna is rotated by the operator, it measures the reception strength of messages sent by the Prey node. RSSI is used as the measure of signal strength, and measured values are recorded as a function of antenna direction, which is retrieved using a tilt-compensated digital compass (yaw, pitch, and roll are all measured).

In works that estimate distance based on RSSI, the statistical variance is notable and stems from numerous sources, some of which depend on the scenario, e.g., the position of the lost mote’s antenna, or whether it is in high and humid grass. Instead, we only rely on the measured RSSI value to deduce direction information based on the directionality of the Hunter’s antenna.

We do not estimate distance, therefore we do not need a strong (linear) dependence of our metric on distance. What is important is a measure with good stability and sensibility properties as far as direction is concerned, independent of the distance, of the position of the other node’s antenna, or of environmental conditions. RSSI satisfies these requirements. Correlation with distance is still useful to have some vague idea about the distance, but it is not essential for the search.

We search for the absolute maximum, and possibly local maxima, in the measurements as a function of antenna direction to identify the most probable directions. These correspond to the arrival angle of line-of-sight and reflected signals. After these are identified, it is the operator’s role to move around, interpret the results, and finally arrive near enough to actually see the node with naked eyes. Even if objects could hamper
line-of-sight or rocks could create strong reflections, by having a human operator that actually sees and interprets the scene and potential reflective surfaces, the mote can easily be found.

A problem we must solve is to solicit the Prey to send enough messages while the antenna is being rotated. For this reason, the Hunter periodically sends special ping messages destined to the selected target node.

A ping message is a standard IEEE 802.15.4 message, crafted to force a response from the target node. First of all, the A (ACK request) flag is set in this message. This allows us to ping almost any standard-compliant mote (exceptions are those that forcibly disabled acknowledgements). Note that the ping is a dummy message, in the sense that only the ACK request control flag matters, its content can be anything. We use empty messages with a pre-selected message ID in order to keep the message short and thus maximize the chances of an error-free reception.

Second, if the software running on the mote includes the Prey component, it also recognizes the message type and actively responds with an application level pong message. Thus, in this case, two responses are sent: first a MAC level ACK is sent by the radio chip or by TinyOS, and immediately after another response is sent at the application level.

We emphasize that the Hunter can search for any mote, even one that was not originally meant to be searched by it. Including the active Prey component in deployed motes, however, has some advantages:

- In most implementations, IEEE 802.15.4 acknowledgements are sent with the last used (or the default) TX power level. This means that if the Prey’s software was set to use a low power level, the Hunter may have problems in hearing the response. The Hunter has better reception than other motes thanks to its high-gain directional antenna, but low TX power levels may reduce the possible ping distance.

  When the active component is embedded in the Prey, the latter recognizes the ping attempt and temporarily sets its TX power to maximum. The original TX power level is saved and restored right after sending the pong message, to leave the original application intact.

- Another feature that could compromise the Hunter’s ability to ping a mote is LPL (Low Power Listening). In LPL mode, the radio is turned off for most of the time and turned on periodically only for a short time window. The active component can be used to temporarily disable LPL to facilitate the search. The original values of LPL are reset after a short time, configurable at compile time in the Prey component.

IV. HARDWARE ARCHITECTURE

The Hunter can be used in two different hardware configurations: full and standalone.

The first one provides all the bells-and-whistles of the tool. It requires a mote, a directional antenna (Section VII-B provides more details on available choices), some Java-capable host device\(^2\), and a digital compass attached to the host.

This configuration is powerful, yet it requires one to carry a computer around, which might be impossible in some situations, e.g., while climbing on rocks to reach a mote dropped by an animal. Therefore, we have also developed the standalone configuration, which serves both as a lightweight option on difficult terrain, as well as as a backup option when a computer or a compass is not available. Although the functionality is somewhat degraded in the standalone case, this operation mode only requires a mote with a directional antenna running the Hunter code, and nothing else.

A. Full configuration

Fig. 1. A Hunter node equipped with a 14 dBi directional patch antenna.

In this mode, the device is assembled from the following components (see Figure 1):

- A host (laptop): it powers all other components, runs the GUI, controls the mote, acquires and visualizes the measurements.
- A mote, attached to the laptop with USB or serial port: it can be any mote based on the TI CC2420 radio chip that supports TinyOS 2.x and is equipped with an external antenna connector. During our tests, we have used Telos-equivalent motes for this purpose.
- A directional antenna, connected to the mote: the directionality of the antenna is essential for the search. It can be fixed to the laptop (as in Figure 1), or it can be attached to a pole for easier handling.
- A digital compass fixed to the antenna, connected to the laptop with USB or serial port. We use a relatively cheap external compass device (the CMPS10 tilt-compensated compass module based on a 3-axis magnetometer and a 3-axis accelerometer \([3]\)). Another option would be to connect the compass to the mote itself. Overall, the use of a stand-alone compass allows the greatest flexibility in the hardware configuration.

Ideally, the compass should be positioned far away from the antenna to ensure that magnetometer readings are not distorted by the magnetic field induced by transmission. It is however easy to find positions (like the one of Figure 1) where the compass works reliably.

\(^2\)The Java GUI is currently tested on Linux, but it should also run on Windows and on OS X, and – with some adaptation – on tablets and other Java-capable devices.
B. Standalone configuration

In the standalone configuration, only a mote and a directional antenna are used. As Figure 2 shows, it is compact and can be build even with a DIY (do-it-yourself) antenna, using only the usual mote, a plastic stripe and some wire.

The downside of this configuration is that the user interface is quite limited: only 2 buttons (of which one is hardcoded reset) and 3 LEDs are available, at least on a Telos. Thus, search is a bit more cumbersome, but our tests show that it still works.

We emphasize that this operation mode is meant only for emergencies on the field (e.g., laptop not available or with battery depleted) or when the terrain is too difficult for walking around with a laptop. In normal cases, the full configuration should be used.

V. SOFTWARE ARCHITECTURE

The Hunter functionality is implemented partly on the mote in nesC over TinyOS, partly on the host machine using Java. Contrary to the usual designs where the mote only serves as a dumb bridge of application layer messages or layer two frames between the serial and the radio link (e.g., in the TinyOS BaseStation application), we implemented part of the program logic on the mote side. This follows from our design requirement to provide backup options for extreme situations, therefore supporting the standalone configuration.

The two parts of the software communicate using TinyOS active messages on the serial link. Measurements are continuously sent in report messages from the mote to the host, while configuration commands are passed in the other direction. The latter include setting the radio frequency, the target node ID, the operation mode, and some timers.

A. Mote Software Architecture

The Hunter mote application consists of the following components:

1) Pinger: This core component of the Hunter periodically sends out messages destined to the selected target node. It waits for the response (both MAC- and application-level responses in case the Prey component is embedded in the target node) and measures its reception strength. After a response is received by the Hunter mote, a measurement report is generated and sent to the host.

In standalone configuration, the Pinger continuously compares new measurements with old ones and records the maximum RSSI value. It also drives the three LEDs as follows: at reception, it displays the difference of the actual value and the recorded maximum as a 3-digit binary number, lighting up all 3 LEDs when the maximum is reached, and showing no LEDs at -7 dB or less. If the operator changes position (going in the wrong direction), the maximum value might need to be reset with the button.

This simple interface is sensitive enough (1 dB), and at the same time has enough dynamic range (7 dB) to indicate the direction.

2) Sniffer: The Hunter also includes a sniffer component that continuously listens on the selected radio channel for messages in promiscuous mode. Whenever a message is seen, reception strength is measured and a report is sent to the host, together with the ID of the node. The node ID is also added to the list of active nodes in the vicinity.

This list, stored on the mote itself, is important when the Hunter is used in standalone configuration. In this mode, instead of entering the 16 or 64 bit ID of the Prey (which would be difficult due to the limited user interface), we allow the operator to cycle through the list of nodes discovered by the Sniffer. Although the items of the list cannot be visualized, in typical scenarios the operator should be able to find the item belonging to the lost mote by rotating the antenna and excluding known nodes.

3) RX power monitor: IEEE 802.15.4 compliant radio chips should be capable of measuring the radio RX power at any time, even when no message is received, in order to implement CCA (Clear Channel Assessment). The CC2420 chip is capable of reporting this value to the microcontroller, and thus to our application. The value is updated once every millisecond, based on which we generate an aggregate report with the average and maximum value every 100 ms. Reports about these power levels complement the set of our other measurements, providing useful information about noise in the given radio frequency. They can also be used in the extreme case of searching for motes that generate non-compliant IEEE 802.15.4 messages either because they are older than the standard itself, or e.g., because their firmware is faulty.

4) Controller: The controller has three important features. First, it receives, interprets, and implements commands from the host. Second, when the Hunter is used in standalone mode, it provides a small user interface (using the 3 LEDs and the 2 buttons available on the mote) to control the operation of the Hunter and switch between various operation modes. Third, it handles the list of active nodes and the selection of the target node.
B. Host Software Architecture

The host runs a Java application that:

- provides data fusion among measurements arriving from various information sources, i.e., RX power from the mote’s radio and antenna direction from the compass\(^3\). Radio measurements are simply augmented by 3-DOF directional information (yaw, pitch, and roll), based on the measurement timestamp.
- includes a GUI providing an intuitive visualization of the measurements, both as a function of time as well as as a function of antenna direction.
- provides inputs for controlling the Hunter mote.

After combining measurements, each data point contains information in six dimensions: yaw, pitch, roll, timestamp, RSSI and the source ID. The latter is either the ID of the node being pinged, the ID of the source of the message caught by the Sniffer, or a special entry meaning that the measurement is from the RX power monitor. The following section describes how this multi-dimensional data set is visualized in the Hunter.

VI. OPERATOR VIEW

The Hunter Java application presents the screen as shown on Figure 3. It is composed of a measurement list (left), a panorama view (middle), a time view (top), and a configuration panel (bottom).

The measurement list presents a projection of our six dimensional dataset, emphasizing the source ID dimension. The time view puts the focus on timestamps, while the panorama view tries to capture two of the three directional dimensions of the dataset.

\(^3\)Positional data from a GPS could also be integrated as a third component of our measurement instances.

A. Measurement list

On the left side, a dynamic list of measurements and associated color codes is shown. The list is continuously populated as new nodes are discovered by the Sniffer or by the Pinger module. The list contains the following entries:

- \(N^{RCV}\): One row for each mote (identified with its node ID \(N\)) discovered by the Sniffer module.
- \(N^{ACK}\): A row for each mote that responded to the Pinger module.
- RX Power AVG: A row for the average signal power received on the selected frequency.
- RX Power MAX: A row for the maximum reception power on the selected frequency.
- Yaw, Pitch, Roll: rows for direction measurements from the digital compass.

The list, besides providing an overview, also provides controls to select the target node and to select which rows to visualize in the views.

B. Views

There are two ways of visualizing measurements: the time view and the panorama view. Both are updated in real-time, facilitating a quick and successful hunt.

1) Time View: Positioned at the top of the window, this view shows all (selected) measurements as a function of time. It could also show direction measurements, thus providing corresponding RSSI and direction values on the same plot.

![Example of time view after rotating the Hunter four times.](image)

Figure 4 demonstrates how the time view looks like when the operator turns around four times. The blue line shows yaw (the sawtooth-like line showing how horizontal direction changes between 0 and 360 degrees), while the black points show Pinger measurements. Note how RSSI values repeat almost the same pattern in each turn, clearly indicating the direction of the node. The brown and pink lines show that we have slightly increased pitch between the turns, while we have kept the antenna horizontal (no roll). Finally, notice how green dots (RX power MAX) indicate the right direction as well, although these measurements are far less precise than those obtained by the Pinger.
2) Panorama View: This is the main view used for the hunt, and therefore occupies the center of the Hunter screen. It presents the reception power from the Prey (ACK measurement) as a function of yaw and pitch (horizontal and vertical angle). In other words, it presents a 360° panorama picture of where the mote could be. Brighter (green) colors indicate stronger signal levels, while darker (red) colors mean weaker measured values. The view is rotated as one turns the device.

Each view has its own configuration panel that can be activated by clicking on the view itself. Of particular interest is the configuration of the panorama view, which presents information aggregated along several dimensions. First, bins are formed according to both yaw and pitch. These bins contain a time series of measurements in the given direction, each measurement still having a timestamp, a roll and an RSSI component. These values are then aggregated into one value, that is then converted into an intensity, thus to a color. The aggregation uses weighted averaging, where the weight has an exponential decay with elapsed time. Weight can also depend on roll, e.g., favoring horizontal antenna positions in order to avoid errors due to antenna polarization. Finally, values are converted into colors using a dynamic color scale that has the minimum and maximum measured values at its extremes. Configurable gamma correction is also used to emphasize differences on screen.

Correctly choosing visualization parameters (bin size, time decay, roll sensitivity, gamma) is essential for providing good visualization in adverse conditions. All parameters are configurable and updated in real time on the view.

![Panorama view with different data aggregation policies](image)

Figure 5 shows how these configuration settings can be used to fine-tune the view. For better rendering, we have inverted the color scheme on this figure: black represents the strongest signal, while white means weak signal. The three plots are all generated from the same dataset. By tuning bin size and gamma, the direction of the Prey (towards North in this case) can easily be highlighted.

C. Main configuration panel

The main configuration panel is displayed on the bottom part of the screen, and includes the following controls:

- frequency selection: selects the frequency channel (11-26) in which the Prey operates. We assume that this is known to the operator. If not, the channels should be searched manually first.
- target mote ID selection: selects the mote to look for. This mote will be probed actively by the Pinger module with high frequency. Note that signals are picked up by the Sniffer module from all the motes operating in the given frequency band, even if the target one is different. A special broadcast target ID (0xFFFF) can also be used to probe all motes in range.
- clear button: to restart measurements.

VII. Evaluation

It is not our goal to provide a detailed quantitative evaluation of the tool’s precision, which mostly depends on the used antenna radiation pattern, usually available from the specifications. Instead, we assess whether our simple tool satisfies our goal of helping the search for lost motes in various environments.

A. Indoor performance

Although our target environment is outdoors, we start our evaluation with measurements from an indoor environment, as this is more accessible and actually more challenging from a communication standpoint. Figure 6 shows the map of our offices where we have “lost” a mote under a table (turned on, of course). The mote is a 3MATE [5], a Telos-equivalent with the usual integrated reversed F type antenna. We have carried out measurements in several points of our building. Four of these (2 inside and 2 outside of the building) are marked on the map. Figure 7 shows the measured signal strength, as a function of yaw. Note that both the map as well as all figures are oriented towards North. The figure also shows the pattern except for DIY antennas, which have largely varying characteristics anyway. See [4] for an evaluation of a cantenna and a tinfoil cylinder antenna.
of the antenna we have used for this experiment: a HyperLink 14 dBi 2x2 array patch antenna.

Although none of the measurement points had a clear Line-of-Sight (LoS), the maximum power vector almost always points in the right direction. An exception to this is point 3, where the direct signal was blocked, most probably by the walls of the staircase. Only the signal reflected from the wall of the internal court is captured. After walking a bit in the indicated direction, measurements from point 4 reveal the correct direction.

The only point where the measured signal strength diagram resembles the antenna pattern is measurement point 1. Even here, the corridor should deform it slightly, tunneling reflected signals, but there are no big reflective surfaces behind or blocking walls in-between. In the case of point 2, the entrance door to the East was reflecting considerable amounts of radio waves (not enough to modify the direction of the maximum signal). At point 4, the main lobe and side lobes are there in the right direction, but superposed with a smaller power reflection from the wall to the North of the observer. Note that the radius of the diagrams show a range of 40 dB, while the usual definition of beamwidth considers 3 dB. Differences of 1-2 dB can already be seen on the GUI.

Even if the indoor environment is much more challenging for our searching task than outdoors (no LoS, strong multipath, huge radio background noise in the ISM bands), our tool turned out to be useful in the hunt for the lost mote.

### B. Outdoor performance

![Prey](image1) ![Prey](image2)

Fig. 8. Scene of the outdoor measurements (left) and the Prey “hiding” in the undergrowth (right).

Next, we have evaluated performance in a typical outdoor scenario—our initial motivation. The left side of Figure 8 shows the meadow where we have carried out measurements, taken from the point of measurement. The mote was hidden is the surrounding bushes, about 60 meters to the East from the Hunter, placed on the ground. The right side shows the node, to which we attached a dipole antenna this time. The mote can barely be seen from the distance of a meter!

We have carried out measurements with three types of antennas: 1) a 14 dBi Yagi antenna; 2) a 14 dBi 2x2 array patch antenna; 3) a DIY Yagi antenna with unknown characteristics\(^5\).

Figure 9 shows both the antennas and their respective measurements. All antennas were easily detecting signal in the right direction. The first two have a very similar pattern, although the Yagi showed the direction slightly better. Still, we think the patch antenna is better for the task, since the Yagi is sensitive to the polarization of the Prey’s antenna. For instance, if the antenna of the Prey were horizontal—which could easily happen when a mote falls down from its intended position—the Yagi would have difficulty catching the signal if it is not rotated (roll) of 90°. The patch antenna is not sensitive to roll, simplifying the task of the operator.

The home-made Yagi antenna appears to have a lower gain (4-5 dBi less than the 14 dBi professional antennas), a narrower beamwidth, and some relatively strong side-lobes. Nevertheless, in practice it worked very well, much to our surprise.

### VIII. Potential Extensions

Several ideas on how to improve the search tool have surfaced during the tests and while writing this paper.

\(^5\)We built the DIY Yagi antenna using a normal dipole antenna as the driven element, adding one reflector element behind it and three director elements in front of it, all mounted on a plastic stripe. The size and distance of elements were tuned (although not precisely) for the 2.4 GHz operation range.
Time-of-flight based range estimation could be added, although precision is limited by the low frequency clock source available on the motes for timestamping the packets. Nonetheless, precision can be improved by techniques such as the one presented in [6].

A collaborative positioning module could also be added to the Hunter and to the Prey component. If there are other nodes in the vicinity, their Prey components could overhear transmissions and help in the localization by sending this data to the Hunter.

GPS could be added to the Hunter, and measurements taken from multiple positions could be combined into a virtual multi-anchor positioning method using the statistical techniques described in [7].

Of course if a GPS is available both at the Hunter and on the Prey, and both of these have clear signal, their relative position can largely simplify the search. Even if the position estimate retrieved by the GPS is largely approximate due to e.g., dense vegetation, it could be added to the pong message and displayed on the GUI to facilitate the initial part of the search. Then, the Hunter’s panorama view can be used at close distance to find the mote.

MOTEHUNTER works only if the mote battery still contains enough power to respond to the ping messages. To avoid polluting the environment with dead motes and batteries, motes could actively monitor their battery state and switch to a sort of “pick-up mode” where, after notifying their state, they switch to a low-power mode and wait for a Hunter to come retrieve them.

Finally, a version that is somewhere in between the standalone and the full configuration could be created, solving the most restrictive issue of the standalone configuration, i.e., the lack of a display. A small LCD (e.g., a HD44780-based module) could be mounted to provide more feedback than what allowed by the 3 LEDs.

Although all of the above are intriguing additions, we found that the tool works quite reliably and effectively as is, without further sophistication.

IX. Related Work

A large body of scientific literature on position estimation techniques for wireless sensor networks deals with collaborative techniques, where pairwise measurements between nodes and measurements from anchors (nodes with fixed and known positions) are used to derive the position of all the nodes. The works in [2] and [7] provide a good overview of techniques and algorithms used in this context. Our case is different from these, as we try to locate one node, using another one which is carried around.

The most similar to our approach is probably the work presented in [4]. Here, although the authors outline the high-level idea of a tool similar to ours, they then focus entirely on the evaluation of two DIY antenna designs and their potential use in mote-finding applications. The two papers complement each other: like many other type of directional antennas, these antennas can also be mounted on the MOTEHUNTER tool.

The work in [6] studies ranging techniques in WSNs using COTS hardware very similar to ours. Although the authors are not dealing with localization, their work could be integrated into MOTEHUNTER to provide even more tools to the operator.

Similar to our tool in functionality, but different in design and usage scenario, is the avalanche beacon (also known as ARVA), used to find people buried under the snow in an avalanche. Avalanche beacons are special purpose commercial tools that operate in the 457 kHz frequency range, with largely different electromagnetic characteristics than our operation range of 2.4 GHz. Modern transceivers use multiple antennas and DSP processing to derive direction and range information [8].

Animal tracking is another field where “beacons” are being used in some deployments [1]. However, contrary to our solution, a dedicated VHF radio (and battery) is typically mounted that emits signals periodically, specifically for the purpose of localization.

X. Conclusions and Future Work

We presented MOTEHUNTER, our solution to the problem of searching for motes in-field. The tool, which has initially been developed for a specific project, is applicable in almost any WSN, helping field work when motes are lost in deployment or during operation.

We described the hardware and software architecture of our tool in detail, and evaluated its performance both indoors and outdoors. Results show that MOTEHUNTER can locate motes in dense vegetation from 60+ meters, where motes can hardly be seen even at very short distances.
XI. ACKNOWLEDGEMENTS

We would like to thank Renato Lo Cigno and Francesca Cagnacci for their comments on an early draft of this paper, and Giorgio Fontana for building the DIY Yagi antenna.

The activities described in this paper have been partially funded through the EU Cooperating Objects Network of Excellence (CONET—FP7-2007-2-224053).

REFERENCES


