A WSN-based, RSS-driven, Real-time Location Tracking System for Independent Living Facilities

Pawel Gburzynski[1,2,4], Wlodek Olesinski[2] and Jasmien Van Vooren[3]
[1] *Vistula University, Computer Science, Warsaw, Poland*
[2] *Olsonet Communications Corporation, Ottawa, Ont., Canada*
[3] *Alphatronics, Lokeren, Belgium*
[4] *Sendronet, Wierzchowo, Poland*
{*pawel, wlodek*}*@olsonet.com, jasmien.vanvooren@alphatronics.be*

Keywords: Wireless Sensor Networks, Real-Time Location Tracking, Ad-hoc Networking

Abstract: We present an indoor, real-time location tracking system (RTLS) based on a wireless sensor network (WSN) where the received signal strength (RSS) readings collected by immobile nodes (Pegs) from mobile (tracked) nodes (Tags) are translated into location estimates for the Tags. The process employs a database of samples previously collected from known locations; thus, the scheme falls into the category of profile-based solutions, with RSS readings being the only kind of input to the estimator. Compared to other schemes hinged on the same general idea, the novelty of our approach consists in systematically taking advantage of multiple transmit power levels at the Tags. This allows us to effectively emulate RFID-type of operation, when a nearby Peg can authoritatively identify the location by perceiving a weak signal from the Tag (indicative of the Tag's immediate proximity), while otherwise falling back to elaborate fitting of multiple readings (collected by several Pegs) to produce a (possibly approximate) location estimate. The location service of our network is an add on to its other duties which consist in providing connectivity within an independent living (IL) facility for the purpose of inconspicuously monitoring the patients, detecting anomalies, signaling alarms, and so on.

1 INTRODUCTION

The work reported in this paper is the offspring of an earlier (and purely academic) study (Haque et al., 2009) evolved into a collaborative effort undertaken by Olsonet[1] and Alphatronics[2] aimed at creating a comprehensive WSN to outfit a number of IL facilities in Belgium. Because of the size limitations, we focus here solely on the location tracking component of the WSN; however, the *primary* role of the network, and the natural need to accommodate the location tracking functionality as an add-on (as opposed to creating a separate system exclusively for that purpose) has had a significant impact on that functionality. This makes our system somewhat different from the popular commercial (and academic) solutions in the area (Deak et al., 2012). One of its characteristic features is the ad-hoc WSN base built around a number of rather idiosyncratic prerequisite solutions (Boers et al., 2012; Gburzynski and Olesinski, 2008). The primary goal of the WSN is to provide an open-ended, low-bandwidth, self-contained, flexible communication platform for detecting and signaling various events, usually related to the well being of the IL patients, e.g., see (Boers et al., 2010). Some of those events can be detected by specific physical sensors (e.g., accelerometers, thermometers) in the Tags carried by (or attached to) the patients. Some other events can be recognized by sensors in the Pegs (e.g., infrared-based motion detectors, ultrasound-based proximity sensors). Yet another class of events consists of those expressible as correlations of measurements carried out by multiple nodes and includes, e.g., various "exceptional" configurations of people and/or objects appearing in the neighborhood (definable in terms of their proximity). We say that such events are detected by *virtual sensors* implemented in the distributed program executed by the WSN nodes, often in collaboration with an external program (executed in a computer controlling the network) dubbed the Operational Support System (OSS). A location tracker is thus an example of a virtual sensor. We expect to see more such virtual sen-

[1] http://www.olsonet.com
[2] http://www.alphatronics.be

sors as the functionality of our WSN is extended to various areas of (medical) diagnostics, rehabilitation, or health-care process monitoring (Fernandez-Llatas et al., 2015).

One may legitimately question the relevance of ad-hoc (*multi-hop*) communication in a facility that comes naturally outfitted with a lot of wired (and wireless) infrastructure, e.g., Ethernet and WiFi. In our view, ad-hoc communication plays an important role in such systems, and its defense can be substantially stronger than the simple observation that it makes the WSN independent of the infrastructure (and thus more reliable). For one thing, the infrastructure independence can be stressed as something more important than simply making the system less prone to power failures or disasters. One of its more tangible advantages is in making the system portable, so, for example, it can be taken "on the road" or quickly deployed (in an ad-hoc manner), e.g., for a special (external) event. Besides, being able to fully control all the communication devices contributing to the WSN makes it easier for the deployer to implement various interesting kinds of *virtual sensors*. The reasonably accurate location tracking service, which we have recently added to the system as a purely logical feature (no extra hardware, no modification of the internal communication scheme) adds substance to this claim.

1.1 A brief overview of the popular approaches to RTLS

We are primarily interested in RF-based techniques, so we ignore here solutions taking advantage of other types of signals (Kaddoura et al., 2005). The most interesting (from our point of view) classification among the former concerns the requisite properties of the RF signals translating into the necessary capabilities of hardware. Needless to say, a viable commercial product must trade in the complication of hardware, not only in terms of pure monetary cost of the devices, but also the cost of their maintenance, including long-term availability of the equipment, integration, adaptation, extension. RSS-based approaches belong to the most conservative end of the spectrum, because practically all RF modules are equipped with some indication of the received signal strength. Unfortunately, those approaches are also deemed least reliable, because the perceived signal level at the recipient depends on many accidental features of the environment, like the (intermittent) configuration of obstacles (including human body) or the relative orientation of the two antennas.

Early attempts to transform RSS readings into locations (Christ et al., 1993) were based on interpreting RSS as a function of distance between the transmitter and the receiver. To this end, a channel model was built where signal attenuation (calculated from the RSS indication) was expressed as a function of distance. While some generic models of this kind have been claimed to well capture the statistics of "typical" indoor environments, e.g., for simulation (Martínez-Sala et al., 2005), their blanket application in RTLS has not met with success. Thus, if used at all, they were augmented with some heuristics or tweaks, accounting at least for some standard obstacles of a permanent nature, e.g., walls (Christ et al., 1993).

The whimsical nature of RSS indications inspired approaches based on other features of RF signals. The most popular among them can be categorized as: Time of Flight (TOF), Time of Arrival (TOA), Time Difference of Arrival (TDOA), Angle of Arrival (AOA), and Near-Field Electromagnetic Ranging (NFER). The first three among them (Deak et al., 2012) require a precise measurement of the propagation time between the sender and the receiver (in a few slightly different flavors), from which the line of sight (LOS) distance can be inferred or, more generally, some relationships between a number of such distances, which in turn can be used for triangulation. With AOA, the measured entity is the angle from the receiver towards the sender; the scheme is often combined with TOA (Venkatraman and Caffery Jr, 2004) to mitigate the impact of obstacles. The NFER technique (Schantz, 2007) relies on measuring the difference in phase between the electric and magnetic components of the electromagnetic field (at the receiver) which tends to depend on the distance. The dependence is pronounced for short transmit antennas, in comparison with the wavelength.

All the above techniques require specific hardware features which are not needed for "normal" RF communication and often get in the way. This basically means that an RTLS must be devised and deployed as a separate system using its own dedicated hardware. For example, NFER devices operate within the AM band (530-1710 kHz) which is exotic from the viewpoint of contemporary *communication* networking. The most successful time/angle-based devices operate in Ultra Wide Band (UWB), to mitigate the effect of reflections, and, at least in our experiments, exhibit significantly lower communication range (combined with a much higher power consumption) compared to popular ISM-based modules. All of them work reliably within LOS environments, where they are truly unbeatable in terms of accuracy, but any obstacles (especially those causing unbypassable reflections) tend to significantly affect the readings. For illustration, we have recently investigated the DW1000 module by DecaWave (Kempke et al., 2015).[3] The device offers practical sub-10 cm accuracy of distance measurement within a clean LOS setup, e.g., with the anchors mounted at the ceiling, thus offering undisturbed paths to the Tag; however, the presence of any obstacles, especially metal objects or a human body significantly affects (increases) the path thus confusing the distance estimate. In some cases, the impact of obstacles can be compensated (Geng et al., 2013), but reliably accounting for walls or ceilings is practically impossible.

1.2 The prerequisites and the problem

The WSN constituting the basis for our system consists of two types of nodes which will be referred to as *Tags* and *Pegs* (Gburzynski and Olesinski, 2008). Pegs are the anchors, i.e., their locations are basically fixed: any change of a Peg's location (as well as the removal/addition of a Peg) involves a nontrivial and clearly identifiable action, while Tags are mobile; these are the devices to be tracked. Both node types are built around the same RF module which is CC430F6137 by Texas Instruments representing the popular CC1100-based family (Texas Instruments, 2014). The device operates within the ISM band (which is 816 MHz for the system at hand).

The Pegs jointly form the actual ad-hoc network whereby (relevant) packets are forwarded, using TARP (Gburzyński et al., 2007), to a central sink dubbed the *master*. The nominal baud rate for RF-communication is 38,400 bps. The maximum (payload) packet length is 54 bytes. The master interacts with the OSS over a USB connection to a workstation computer. A Tag communicates with the system via Pegs. There is no presumed (even momentary) association of a Tag to a specific Peg for communication with the network; however, some Tags have subsets of their functions restricted to the neighborhoods of their "local" Pegs. Conceptually those functions correspond to remotely controlling some equipment within the local room or apartment. There exist several types of Tags, but the difference is irrelevant for the problem at hand. The simplest Tag variant acts as a panic button triggering alarms delivered to the master and conveyed to the human operator. This is also the most obvious example of a case when location tracking comes into play.

The location tracking problem is defined as attributing Tags (worn by people or attached to some objects) to loosely described locations. Note that, unlike (Haque et al., 2009), by *location* we do not understand a point, say in 3-d, to be pinpointed with some desirable accuracy, but a named area (or space) whose size and shape generally depend on the context. For example, its granularity may correspond to a room, or even an apartment or a corridor, if it happens to be satisfactory from the viewpoint of the operator. Generally, the system is not expected to be 100% reliable. In particular, it may offer a number of alternative (suspect) locations ranking them according to some measure of likelihood.

Normally, a Tag is dormant most of the time. It may carry out some variant-specific functions, but generally it is under no obligation to report more often than, say, once per hour generating a dummy "alarm" to manifest its being alive (and, e.g., indicating the status of its battery). The energy budget of a Tag is usually tight. The device is expected to last for several years without replacing the battery. The current consumption of a dormant device is of order 2-4 μA. Any radio activity (transmission or reception) raises this figure to about 15 mA, i.e., by four orders of magnitude.[4] Consequently, minimizing the duration of Tag duty cycles is one of the system's primary goals.

Note that the above assumptions shed a new light on the comparison of the RSS-based approach to location tracking with the other techniques mentioned in Sec. 1.1. For example, if we switched to a TOA-capable RF-module, we would not be able to take the full advantage of its capability, because the application where such a device truly excels, i.e., locating the Tag precisely within a (potentially large) room outfitted with several (at least three, and preferably four) anchors-Pegs deployed within the Tag's LOS (say, in the corners at the ceiling), is of no interest to us. Instead, we want to be able to tell apart the different rooms (or blocks of rooms) for which the TOA approach doesn't work too well (if it works at all), because of the impossibility of LOS communication. The Pegs might still be able to act effectively as RFID readers, locating the Tags by sheer proximity, but the TOA capability becomes then a messy overkill. Thus, we would hope instead that the kind of localization service envisioned for the project can be accomplished with a simple RSS-based approach, assuming some reasonable coverage of locations by Pegs, preferably, no more than one Peg per apartment (as in the original edition of the underlying WSN).

[3]http://www.decawave.com

[4]The supply voltage is 3 V. This level of power consumption is typical and representative of microcontrolled ISM RF modules.

2 THE TRACKING SYSTEM

2.1 The idea

A tracked Tag emits at its discretion so-called *location bursts* which are series of short packets sent back-to-back at different power levels. In the present system, a location burst consists of 32 packets, 4 packets per each of eight discrete transmit power levels numbered 0 through 7. Thus, the first four packets of a burst are sent at power level 0, the next four at power level 1, and so on. All packets in a given burst are marked with the same sequence number (allowing the recipient to tell that they belong to the same burst) as well as with individual indexes (indicating their position within the burst). Any Peg receiving any packets of a burst prepares a *location report* and expedites it to the master.

The transmit power levels have been calibrated experimentally, such that the difference between two adjacent levels is about 5 dB. The value of RSS directly translates into the received power level in dBm quantized into steps of 0.5 dBm (plus some fixed offset). We do not assume that the location bursts are issued at some specific intervals: the Tag emits a burst when it is considered prudent. For example, pressing a panic button will cause a burst (followed by the event packet), because it is desirable to locate the Tag at this moment. Note that, generally, the intervals between bursts can be long, so we do not incorporate history (previous estimates) into the location tracking algorithm. It is obvious that making the bursts periodic (and frequent) and accounting for the history of recent estimates will tend to improve the accuracy of estimation. For example, (Ni et al., 2004) discuss a remarkably accurate scheme achieved with just one anchor node (a WiFi access point) plus some profiling, some modeling, and a rather heavy reliance on the history of previous estimates. Another possibility is to apply prediction models for traffic (Cho and Kwon, 2016) which, considering the specifics of the target environment, is likely to further reduce the uncertainty in position. This work has been left as a natural goal for the future, the present focus being on maximizing the yield from the pure memory-free (and context-free) scheme.

The only relevant kind of data arriving at the location server is a sequence of reports from Pegs relating to the fragments of the bursts they have managed to receive from the tracked Tags. Formally, a report is a 4-tuple $R = (P, T, r, V)$ where P is the Peg Id, T is the Tag Id, r is a by-Tag serial number identifying the burst, and $V = (v_0, \ldots, v_7)$ is the RSS vector consisting of eight unsigned single-byte values representing the averaged RSS readings for the eight power levels (note that a Peg may receive up to four packets per level). A value of zero (which can never be a legitimate RSS reading) means that not a single packet corresponding to the given power level has made it to the Peg.

The tracking is driven by a database of samples collected from known locations. For this, the area is profiled and the reports collected during this process are stored in the database along with their locations. During the actual tracking, the location reports are compared against the profile samples. Roughly speaking, the Tag's location is estimated as the one attributed to those profile samples from the database that best match the tracking reports.

2.2 Profiling

Formally, locations are identified by numbers referred to as *internal* location identifiers. The OSS is responsible for mapping those internal identifiers to *external* locations, i.e., descriptors presented to the human operator. A single external location may correspond to a number of internal locations acting as their union. Also, different internal locations need not refer to disjoint subareas of their corresponding external locations. The mapping is intentionally ambiguous and thus flexible. For example, it may happen that a large and geometrically complex external location would have to be covered by a large number of samples for the coverage to be representative and accurate. While looking for the single location best fitting a tracking sample, the server has to consider subsets of profile samples from the candidate locations (Sec. 2.3.2). A partitioning of a larger location into smaller fragments may reduce the complexity of the problem while also providing for more representative coverage of the smaller and simpler internal areas with their individual sets of samples. On top of this, overlapping internal locations may represent the same areas sampled in some particular way, as to capture some modal feature of the Tag, e.g., its particular elevation above the floor, orientation, proximity to the body. While looking for best matching collections of samples, the server may identify different internal locations which will be mapped by the OSS to the same external one. The way to look at it is that internal locations are identifiable containers for collections of samples. Then, those containers are mapped into the actual (external) locations.

The aim of the profiling procedure is to collect samples from known (internal) locations. The way the server guesses at the location of a tracked Tag (Sec. 2.3) suggests these recommendations for sample collection.

1. The (internal) locations should be reasonably simple, preferably rectangular and close to squares; hence the recommendation to split tricky (external) locations into simple (internal) ones.
2. The number of samples per (internal) location need not be large, but the samples should be complete and reliable; consequently, it makes sense to combine/average multiple samples taken from the same point of a given location until a) all the Pegs that can hear the Tag have been included, b) the readings for those Pegs appear representative and complete, e.g. there are no holes (missing entries) in the RSS vector following the lowest-power nonzero entry.

The server offers hooks to facilitate profiling. The user's end of the interface consists of a simple remote GUI to the server (running on a tablet or laptop) and a Tag. Having moved to the point from where a sample is to be collected, the user *anchors* the Tag. This operation tells the server to assume that any burst reports referring to the Tag should now be attributed to a sample. The sample is identified by the location Id and the collection point Id, the latter being a location-relative mark whose purpose is to tell apart different samples collected from the same location. A button press on the Tag will trigger a location burst. A report from that burst arriving at the server (via the master node) will be added to the sample. Multiple reports from the same Peg are merged, i.e., the RSS vector entries of the sample are averaged using this formula:

$$v_{i(j)} = v_i \times \alpha + v_{i(j-1)} \times (1-\alpha) \qquad (1)$$

where $v_{i(j)}$ is the updated value of entry i in the averaged RSS vector, $v_{i(j-1)}$ is the previous value of that entry, v_i is the new reading (from the newly received report), and α is a parameter (with the default value of 0.3). The update does not occur when $v_i = 0$, i.e., the report has no RSS value for the given power level. This exponential moving average ensures that new samples tend to override old ones, so previously profiled locations can be easily (and selectively) re-profiled later. The GUI tells the user when the location bursts cease to perceptibly improve the sample (see point 2 above), so the profiling procedure can proceed to a different point.

Formally, a sample is a triplet $S = (L, s, \{P_i, V_i\}_{i=0}^{k-1})$ where L is the (internal) location identifier, s is the sample identifier (relating to the collection point within the location), and the third element is the set of pairs: Peg Id (P_i) and the corresponding RSS vector (V_i), representing the burst reports that have contributed to the sample (k is the number of different Pegs that have received the bursts).

Note that, except for separating the different samples, the sample identifier s plays no other role. In particular, the server does not care where exactly the sample has been collected (the specific position of the collection point within L). All that matters is that some number of (disjoint) samples are attributed to a given (internal) location. Our present policy is to collect five samples from every internal location, four samples from the (approximate) corners and one from the (approximate) center.

2.3 The estimation algorithm

2.3.1 Collecting reports for estimation

A location report referring to a non-anchored Tag (Sec. 2.2) arriving at the server can be used for estimating the Tag's location. The server stores (stashes) the arriving reports on a per-Tag basis, aggregating them into a *tracking set*, i.e., sufficiently many reports (for the given Tag) to use them for location estimation. Note that reports from different Pegs arrive independently and can be delayed. A stashed report is time-stamped with its arrival time.

The procedure is carried out independently for different tags, so let us focus on some Tag T. Suppose that a new report $R = (P, T, r, V)$ arrives at the server at time t. The server examines the current set of stashed reports for T and, for each report $R_i = (P_i, T, r_i, V_i)$ performs these actions:

1. Let t_i be the time stamp of R_i. If $r_i \neq r$ and $t_i + \delta_s < t$, then R_i is deemed obsolete and discarded from the stash. The default value of δ_s is 4 seconds.
2. If $P_i = P$, then the new report augments the stashed one. The elements of V_i are updated in R_i according to formula 1, the time stamp of R_i is refreshed to t, the new report R is discarded. Otherwise, R_i is retained and skipped. Then the next report from the stash is taken until all the reports have been examined.
3. If all stashed reports have been processed, and P hasn't matched any P_i in step 2, the new reading is added to the stash with time stamp t.

After completing the above loop, the server is ready to attempt a location estimation, provided that the stash contains at least K reports (the default value of K is 3). However, as the server never knows when the last report for a burst is going to arrive, it delays the attempt for $\delta_e = 3$ seconds. To prevent indefinite postponement in the unlikely scenario when reports keep arriving frequently and incessantly, the server will force an estimate when the number of postponements has reached $N_m = 16$ without a single estimate.

The estimation procedure is described below.

2.3.2 The outline

We start with the tracking set, i.e., the current collection of stashed reports $\{R_i\}_{i=0}^{M-1}$, where M is the set size ($M \geq K$) and $R_i = (P_i, T, r_i, V_i)$. Let $v_{i,l}$ denote the l-th element of vector V_i. For each power level l, $0 \leq l < 8$, we construct U_l as the set of tuples $(P_{i_l,l}, v_{i_l,l})$ such that $v_{i_l,l}$ are the three largest elements of $\{v_{i,l}\}$, over all i, $0 \leq i < M$. Then we discard all elements from U_l where $v_{i_l,l} = 0$. Note that U_l consists of at most three elements and it can be empty. In plain words, for each of the eight power levels, we select from all the stashed reports three with the largest RSS readings, along with their associated Pegs.

Subsequently, the estimator executes a loop indexed by the power level l starting at $l = 0$ (the lowest level) and going up to 7. At any iteration, the estimator maintains the current set of candidate locations C consisting of pairs (L, r), where L is a location and r is its inverse rank (or badness). The set is initialized to "all locations" with the identical ranks of 0. Then we proceed as follows.

1. Consider iteration l. Its objective is to calculate a set of candidate locations C_l with ranks based solely on U_l. Then the two sets, C and C_l, are intersected and their ranks combined. The intersection will possibly trim down the set in size, while the new ranks will diversify the likelihoods of the locations still remaining in the set.

   If U_l is empty, then the iteration is skipped which means that the current set of candidate locations C is unaffected and carried over to the next iteration. This is equivalent to setting C_l to "all locations" with the same ranks of 0.

2. Suppose that U_l is nonempty. Then C_l is determined as consisting of those locations that have at least one sample in the database that includes all the Pegs from U_l and a nonzero RSSI entry at level l for each of those Pegs. The procedure for calculating the ranks for C_l is discussed in Sec. 2.3.3.

3. Then an attempt is made to intersect C_l with the running set C. If the intersection is empty (which will also happen when C_l is empty), the loop is exited with the last nonempty C. Otherwise, C is set to the intersection. The new ranks (for those locations that remain in C) are calculated as the sum of the old ones and the new ones (from C_l). For example, if C initially consists of locations 4, 7, 10, 11, 14, ranked 10.0, 8.5, 5.0, 14.0, 3.0, respectively, and C_l includes locations 3, 7, 8, 11, 12, 15 with ranks 2.0, 9.0, 5.5, 1.0, 2.0, 6.5, then the running set C at the end of the iteration will consist of locations 7 and 11 with ranks 17.5 and 15.0, respectively.

When the loop is terminated, the locations in C are sorted in the inverse order of their ranks and returned as the location estimate.

Note that each turn of the loop (for a higher power level) will tend to trim the running set; however, when we get into an apparent "contradiction" (an empty set), we interrupt the process. This way, low power levels have priority in shaping the set of candidate locations (and also in resolving contradictions). A close match to some samples on low power levels is indicative of proximity to the corresponding Pegs, so it can be used a reliable shortcut for estimating the Tag's location. In the most advantageous case, there will be just one location selected this way, approaching (passive) RFID-type operation equivalent to (almost) stumbling on the location's characteristic Peg(s). Should the identification by low power levels turn out to be imperfect, then: 1) the higher levels may help the estimator trim out some locations, 2) a subtler comparison of the reports to database samples will rank the candidates, thus offering grounds for selection.

Note that augmenting the WSN with extra Pegs is relatively cheap: after all, this *is* an *ad-hoc* network, so the extra Pegs can be just thrown in without any configuration. This way, important locations (deserving better accuracy) can be more densely covered by Pegs, which can be done in stages, e.g., in response to perceived problems. While formally such additions should require re-profiling, note that old samples retain their discriminating properties for as long as their "strongest" Pegs remain strong in their locations. Thus, the re-profiling need only be applied to the immediate neighborhood of the added Pegs.

2.3.3 Calculating location ranks

The number of elements in U_l can 1, 2, or 3. The case of zero does not apply to location ranking, because the ranking step is skipped in that case. The first two cases are considered degenerate, so let us start from the last one. The three RSS values in U_l can be interpreted as a point in 3-space (in the positive quadrant). Thus we deal with one point q obtained from U_l and a set of points Q obtained from the database samples (one point per sample) selected in step 2 in Sec. 2.3.2. Those samples are grouped by their locations. Let $Q_L \in Q$ be the subset of Q containing all those samples selected in step 2 of the algorithm that are attributed to location L. The rank of location L

is calculated as the Euclidean distance from q to the minimum convex hull encompassing the points in Q_L.

With the above approach, a sample, including the tracking set, is interpreted as a point in some simple metric space, so it makes sense to talk about the distance between a pair of samples. One can expect that when we take two samples, from two different spots of some location, the RSS readings of the Pegs appearing in both samples will differ, but as one moves between the two spots, the transition between the vectors (points) is likely to be smooth. Locally, in the best of the possible worlds, the transition can be approximated by a straight line connecting the two points, especially if no better model is available. As we can choose the collection points for samples, we can (in principle) try to account for the nonlinearities by selecting the points where the transitions are likely to become messy.

Given multiple samples referring to the same location, the linear approach to interpreting transitions among them boils down to drawing a polyhedron connecting their points. As, generally, there is more than one way to draw such a polyhedron, the most natural (and well defined) representative of them all is the minimum convex hull encompassing all the points. Then, it makes sense to assume that the interior of that hull represents in a certain way the location covered by the samples.

2.3.4 Adjustments

The algorithm can be tuned through a number of parameters. First, the RSS values subjected to the calculations described in Sec. 2.3.2 and Sec. 2.3.3 can be rescaled via an interpolation table, e.g., to amplify differences in large values (which are more indicative of proximity to the Peg and, generally, more reliable). Also, when combining the location ranks (Sec. 2.3.2, step 3), the ranks from different power levels can be multiplied by different factors, e.g., to assign a higher importance to higher levels.

Although Tags are built as fairly homogeneous devices, some discrepancies in the RF properties of different specimens of Tags are unavoidable. This means that RSS readings from different Tags obtained under identical conditions may differ slightly. To compensate for this, when calculating the distance between the tracking point q and the convex hull of Q_L (Sec. 2.3.3), the ranking algorithm may turn q into a segment and calculate the distance from the segment to the hull. Suppose that $q = (x, y, z)$ The segment is obtained as the set of points $Q_s = \{(x+t, y+t, z+t)\}$, $-\rho \leq t \leq \rho$, where ρ is the tolerance parameter settable on a per-power-level basis. This simple trick assumes that any fluctuations in the RF characteristics translate into a linear (base-independent) shift in the transmit power level, which is consistent with the interpretation of RSS as the actual power in dBm (plus/minus some offset). Note that nonzero ρ will tend to blur the quality of location separation by hulls, so it has to be selected with care.

3 EXPERIMENTS

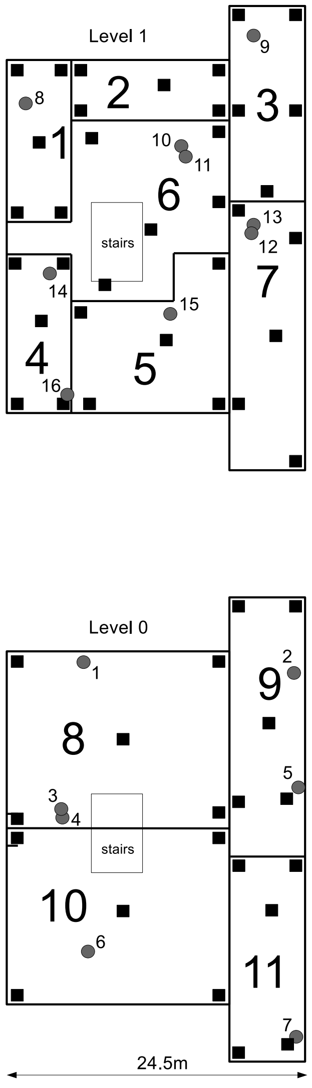

Figure 1: The test deployment.

3.1 The test system

Fig. 1 illustrates a test deployment of our system. The area consists of two floors, one of them (Level 1) comprising office space with irregular rooms separated by walls. Level 0 includes a large (exhibition) hall, taking most of the space on the left-hand side of the floor with some (relatively small) meeting rooms on the south side. Note that the separation of the two levels into *locations* does not strictly follow the division of the area by walls. In particular, only some walls are drawn at Level 1 (practically every location encompasses several rooms), and the meeting rooms at Level 0 (formally falling into location 10) are not drawn at all. Also, the separation of the exhibition hall at Level 0 into locations 8 and 10 is purely imaginary. That was intentional: we wanted to capture (also) the malicious and fuzzy scenarios where the boundaries between location are not clearly demarcated. The location algorithm doesn't explicitly incorporate any model of walls and we expect it to operate (possibly with varying degree of accuracy) for any conceptual division of the monitored area into locations.

The gray circles in Fig. 1 denote the Pegs of our WSN, 16 of them altogether. Their deployment, while not completely accidental, is far from perfect. Note that, e.g., in locations 6 and 8 there are pairs of Pegs located very close together (Pegs 10, 11 and 3, 4). We shall clearly avoid such (pointless) configurations in production systems. In the test network, they have been dictated by the logistics of other tests carried out in parallel with the preliminary tests of the location tracking service (the co-located Pegs are devices of slightly different types). Consequently, some locations are better covered than others. For example, the rather difficult (by its shape) location 11 has only one Peg (7) which, to make the matters worse, is situated in a corner. Thus, one cannot hope that location 11 will be always perfectly identified, because the kind of passive RFID emulation mentioned in Sec. 2.3.2 is only going to work close to one of its corners. All these problems should be OK for tests, as long as we understand the limitations. We can easily tell by looking at Fig. 1 where the location algorithm can be expected to act accurately, and where it is likely to be confused because of the deficiencies in coverage by Pegs.

3.2 Profiling

The black squares in Fig. 1 mark the collection points for samples. Within the confines of physical accessibility, we have tried to collect samples from the four corners of each location as well as from its center. We shall refer to those points as NW, NE, SW, SE, and C. Every such a point translates into one sample (being an average of multiple takes, Sec. 2.2) yielding the total of 55 samples. For illustration, here is the NW sample for location 3:

```
12   0   0   0   0   0  67  75  83
11   0   0   0   0  67  75  78  80
 8   0   0   0   0   0   0  82  82
 9  82  93 103 112 124 135 148 157
10   0   0   0   0  71  72  87  95
 5   0   0   0   0  72  75  80  90
13   0   0   0   0   0  64  74  83
 2   0   0   0  69  78  84  92  99
```

Each row consists of 9 values: the Peg Id followed by eight RSS readings for the power levels from 0 to 7. Recall that 0 means "no reading." Note that the readings for increasing power levels are increasing; also, the magnitude of the RSS values intuitively agrees with the positions of Pegs relative to the collection point.

Owing to the inherent lack of reliability in wireless communication, and because the packets in location bursts are neither acknowledged nor retransmitted, any single report from a location burst can be incomplete. We have to be prepared for this while tracking locations; however, the quality of samples is important, because a missing reading (or an entire vector) may affect the interpretation of all tracking sets. Consequently, a sample is only accepted if it cannot be improved any more by subsequent bursts issued from the given collection point, where by improvement we mean filling in an entry that was absent after a previous take. Also a sample containing holes, i.e., zero entries following non-zero ones, must be improved before being accepted.

3.3 Tracking

A low-level RSS entry in a burst report can only be nonzero when the Tag is relatively close to the Peg. Thus, some areas are easier to recognize than others. For example, here is a tracking set collected from the SW quadrant of location 10:

```
 3  72  77  95 103 113 124 137 145
 5   0   0   0   0   0  70  86  96
 6  91  95 106 117 128 136 149 157
 7   0   0   0   0  80  92 104 109
 8   0   0   0   0   0  80  97 103
10   0   0   0  71  78  92 103 109
12   0   0   0   0   0  67  84  93
13   0   0  65  76  90  96 110 115
16  82  94 101 112 124 133 147 154
```

Recall (Sec. 2.3.2) that the power levels are scanned from low to high, i.e., from left to right. For power level 0, the only three nonzero entries are those for Pegs 3, 6, and 16. As it happens, there are only two samples where 0-level RSS readings for all those Pegs are present, namely SW and C for location 10. Thus, a single location is found in the very first iteration, and the algorithm stops immediately. In fact, the single report for Peg 6 would suffice, because the two samples are the only ones that include 0-level readings for that Peg. If the report didn't make it, then the two remaining Pegs, 3 and 16, would do. For Peg 16, there are matching samples in locations 4 (SE, SW, C), 5 (SW, C), and 10 (SW, C), and for Peg 3, the samples are in locations 8 (SW) and 10, the only location shared by them being 10.

The answer produced by the algorithm is the last set C (see Sec. 2.3.2) presented as a list sorted by the location ranks. Recall that, as calculated by the algorithm (after the last iteration), those ranks are reverse, so they represent "badness" rather than "goodness" of a location. Before presentation, they are transformed into a positive measure of goodness according to this prescription. Let r_i, $i = 0,\ldots,n-1$ be the badness of the i-th location from the list. Let $a = (\sum_{i=0}^{n-1} r_i)/n$ be the average of those values. Let $g_i = 1/(r_i + a)$ and $S = \sum_{i=0}^{n-1} g_i$. Set $h_i = (g_i/S) \times 100$ and return h_i as the goodness measure of location i. Note that all h_i values add to 100 (so they can be viewed as percentages) and lower values of r_i translate into higher values of h_i. We do not worry about the precise meaning of those values, e.g., as probabilistic likelihoods (Röhrig and Müller, 2009), because our algorithm doesn't purport to operate as a rigorous probabilistic classifier. For the above (easy) estimation case, the server returns a trivial list consisting of the pair (10, 100), i.e., location 10 has been identified as a single candidate with the goodness rank of 100. Of course, it doesn't mean that the location is in any sense *guaranteed* to have been guessed correctly, only that, based on the available samples, no other match could be made.

For a more challenging case, here is a tracking set obtained from a Tag within the NE annex of location 5:

```
 5   0   0   0   0   0  70  88  94
 7   0   0   0   0   0  83  97 103
 9   0   0   0   0   0   0  68  76
10   0   0   0   0   0   0  63  75
12   0   0   0   0  73  85  97 106
13   0   0  62  71  83  96 105 109
16   0   0   0   0   0  75  80  96
```

The first non-zero power level is 2 with a single report (Peg 13). In the first stage of the iteration for this power level, all the samples with a reading for Peg 13 and power level 2 are identified. This brings in the following list:

```
 1  < 59>  < 68>  < 68>
 2  < 65>
 3  < 60> < 81>
 4  < 69> < 69> < 58>
 5  < 65> < 91> < 67> < 75> < 72>
 6  < 61> < 78> < 72> < 72>
 7  <109> <103> < 60> < 83> < 60>
 8  < 65>
 9  < 59> < 73>
10  < 69>
11  < 77> < 69> < 74> < 65>
```

The first number in each line is the location identifier; following it, we see the list of points, each point encapsulated in <...>, corresponding to the RSS readings for Peg 13 extracted from all those samples where there was a reading for Peg 13 at level 2. As there is a single Peg for power level 2 in the tracking set, the points are one-dimensional, i.e., there is a single value-coordinate for every point. Note that the algorithm doesn't care about the samples from which those readings come, but it does care about the locations. We can see that the discrimination of locations is far from perfect at this stage: the set includes all locations, because each of them has at least one sample with a reading for Peg 13 at power level 2. In particular, for locations 5 and 7, all five samples include such a reading, while locations 2, 8, and 10 offer just one sample each.

The iteration ranks the locations based on the minimum distance between the tracking point (the value of its single coordinate is 62) to the convex hull encompassing the matched points from all the samples for a given location. In this one-dimensional, degenerate case, the distance boils down to absolute difference, and the hull is simply the range of the respective values. For power level 2, the algorithm assumes the tolerance ρ (Sec. 2.3.4) of 8 units on either side which means that the tracking point is turned into a segment (range) from 54 to 70. This ranking gets us nowhere, because all the ranks turn out to be zero in this metric. Thus, the next iteration basically starts with the clean slate.

For power level 3, we again see a single RSS entry for the same Peg 13. The case is similar to the previous level (not surprisingly, all locations have samples with entries for Peg 13 and power level 3) and, following the iteration, the ranks still remain at all zeros. For iteration 4, there are two entries: 73 and 83, for Pegs 12 and 13, respectively. The coordinates are listed in the increasing order of Peg numbers, so their interpretation as dimensions is unambiguous.

Still, all locations include entries for the two Pegs (so they all still remain in the game) with the list of points:

```
1 < 70  67> < 69  67> <81 65> <77  66> <77 88>
2 < 68  72> < 70  79>
3 < 68  74> < 98 101>
4 < 70  81> < 78  91>
```

```
 5 < 78  80> < 98 109> <70 84> <69  76> <70 89>
 6 < 70  81> < 76  70> <91 98> <78  85> <80 91>
 7 <119 130> <108 120> <70 76> <84 103> <75 78>
 8 < 76  81>
 9 < 68  59> < 73  92>
10 < 73  69> < 81  81>
11 < 86  93> < 77  90> <81 85> <67  81>
```

This time the points are two-dimensional, so the hulls are non-trivial and amount to polygons. The tolerance ρ for power level 4 is 7, so we are looking at distances between the segment $(73+t, 83+t)$, $-7 \leq t \leq 7$ and the polygons built of the above sets of points. This brings in the non-trivial ranks: 5→1.0, 6→1.0, 11→1.0, 7→1.01, 4→1.6, 1→1.7, 9→2.4, 3→3.5, 8→3.74, 2→6.48, 10→7.48.

Power level 5 is the first non-degenerate case with all three coordinates present. The Pegs with the three largest RSS readings are 7, 12, 13 and the tracking point is (83, 85, 96). This time, the list of locations with samples matching all three Pegs at power level 5 consists of Pegs 5, 6, 7, 8, 9, 10, 11, so these are the locations carried over to the next iteration, their new ranks being: 7→1.11, 11→1.71, 5→1.73, 8→4.72, 10→8.48, 6→10.27, 9→14.28. This set of locations remains unchanged through the remaining two iterations, however, their ranks change with the final values (after iteration 7) being: 5→5.81, 11→8.0, 7→10.1, 10→10.02, 8→12.97, 6→33.04, 9→33.06. These (badness) values are transformed into the following (goodness) percentages: 5→19, 11→17, 10→16, 7→15, 8→14, 6→8, 9→8. The estimation does not look extremely reliable, but the top candidate has been guessed correctly.

A meaningful, quantified expression of the results from our (still preliminary) experiments is difficult, mostly because the location tracking problem has been defined in rather subjective, qualitative terms: to have a satisfactory, practical solution separating named locations (potentially of various sizes and shapes), with honest acceptance of failures in those cases where the environment is predictably unfriendly. This is in some contrast to previous work (Haque et al., 2009) where the problem was defined as estimating Cartesian coordinates of points in 2-space (so one could say by how far one missed the target). In the present case, the success rate depends on where the Tag is positioned within a given location, and it isn't easy to express numerically how much more important (or relevant) some of those spots are than the others. For example, location 7 in our test setup is poorly covered by Pegs, so estimates taken from the bottom half of that location tend to be mostly useless, being confused with locations 11, 5, and 10. This is hardly unexpected. On the other hand, 95% of attempts from location 6 succeed perfectly, with the rate approaching 100% in the NE section, with only slight deterioration as one gets closer to the boundaries. Notably, even location 2, which has no specific Peg, is correctly identified (87% success rate in the central area), although the results tend to be worse depending on the proximity to the neighboring locations. This is because the distribution of nearby Pegs provides enough diversity and balance in their RSS readings to transform those readings into meaningful (and mostly correct) location ranks. As the success rates depend on the position within the monitored locations, they have to be weighted by the distribution of Tags in a practical deployment (how likely the tracked person or object is to be positioned close to the central area, as opposed to its boundary) to be meaningful. Such weights can be arrived at by inspecting room (apartment) layouts, i.e., the arrangement of furniture, or even suggesting layouts that will increase the likelihood of successful positioning. Owing to the somewhat accidental distribution of Pegs in our test network (not quite inspired by the location tracking problem) one can be sure that a better crafted design will result in more reliable estimates. Our experiments suggest that having a Peg not far from the central spot of a location will practically guarantee successful positioning in all interesting cases.

4 CONCLUSIONS

We have presented a practical location tracking algorithm to accompany a WSN deployable in an institution where people or objects need to be tracked with the accuracy of rooms or apartments. The flexibility inherent in the ad-hoc wireless network makes it possible to adjust the deployment of Pegs to tune the accuracy of location tracking to the importance (relevance) of different locations.

The known problem of poor (and generally capricious) representation of locations (distances) by RSS readings is addressed in our solution in two ways. First, by diversifying the transmit power levels of packets in location bursts we attempt to emulate passive RFIDs, thus providing for easy and reliable answers in those situations where the Tag happens to be located truly close to a nearby Peg. This is because an RSS reading obtained at the lowest transmit power level is practically always indicative of immediate proximity to the Tag, regardless of any accidental differences in the actual value. When the low-power response appears ambiguous, we apply an iterative ranking scheme whose role is to eliminate some of the candidate locations and rank any remaining ones in the order of their assessed goodness.

Second, we admit simultaneous, multiple "planes" of sampling whereby the same (logical) location can be represented by its (not necessary disjoint) components or aliases. By sampling such aliases under different RF propagation conditions we can incorporate into the scheme potential plethora of dynamic disturbances that the monitored area can be exposed to in a way affecting its representation by RSS samples. Our preliminary experiments suggest that satisfactory performance can be attained without extensive (multi-plane) sampling; however, it helps to hold a few cards up one's sleeve in readiness for potential problems.

The algorithm is parameterized and can be extended in several ways. The dimensionality of points used for ranking locations (Sec. 2.3.3) can be increased, say to 4 or 5. We stopped at 3 in an attempt to strike a balance between the useful information and noise available within a tracking set consisting of many reports. Three reports appear to carry enough information to try a location estimate. With more reports available, it probably makes better sense to focus on selecting their best (strongest) 3-element subset, instead of applying them all.

In its present version, the algorithm keeps no history of previous location estimates for a Tag. One can easily think of affecting the location weights, especially in truly dubious cases, by higher ranks given to the locations being close neighbors of those visited recently. This kind of enhancement may be easy (and probably will be incorporated into the production version of the algorithm), but note that it redefines the problem a bit. At present, the algorithm doesn't care about the geometry of locations, including their spatial relationship. A sensible incorporation of history (or mobility prediction models) will require additional input to the algorithm.

REFERENCES

Boers, N., Nikolaidis, I., Gburzyński, P., and Olesiński, W. (2012). PICOS & VNETI: Enabling real life layer-less WSN applications. In *Proceedings of Sensornets'12*, Rome, Italy.

Boers, N. M., Chodos, D., Gburzynski, P., Guirguis, L., Huang, J., Lederer, R., Liu, L., Nikolaidis, I., Sadowski, C., and Stroulia, E. (2010). The smart condo project: services for independent living. *E-Health, assistive technologies and applications for assisted living: challenges and solutions. IGI Global.*

Cho, H. and Kwon, Y. (2016). RSS-based indoor localization with PDR location tracking for wireless sensor networks. *AEU—International Journal of Electronics and Communications*, 70(3):250 – 256.

Christ, T. W., Godwin, P. A., and Lavigne, R. E. (1993). A prison guard duress alarm location system. In *Security Technology, 1993. Security Technology, Proceedings. Institute of Electrical and Electronics Engineers 1993 International Carnahan Conference on*, pages 106–116. IEEE.

Deak, G., Curran, K., and Condell, J. (2012). A survey of active and passive indoor localisation systems. *Computer Communications*, 35(16):1939–1954.

Fernandez-Llatas, C., Lizondo, A., Monton, E., Benedi, J.-M., and Traver, V. (2015). Process mining methodology for health process tracking using real-time indoor location systems. *Sensors*, 15(12):29821–29840.

Gburzyński, P., Kaminska, B., and Olesinski, W. (2007). A tiny and efficient wireless ad-hoc protocol for low-cost sensor networks. In *Proceedings of DATE'07*, pages 1562–1567, Nice, France.

Gburzynski, P. and Olesinski, W. (2008). On a practical approach to low-cost ad hoc wireless networking. *Journal of Telecommunications and Information Technology*, 2008(1):29–42.

Geng, Y., He, J., and Pahlavan, K. (2013). Modeling the effect of human body on TOA based indoor human tracking. *International Journal of Wireless Information Networks*, 20(4):306–317.

Haque, I., Nikolaidis, I., and Gburzynski, P. (2009). A scheme for indoor localization through RF profiling. In *ICC'09*, Dresden, Germany.

Kaddoura, Y., King, J., and Helal, A. S. (2005). Cost-precision tradeoffs in unencumbered floor-based indoor location tracking. In *3rd International Conference on Smart Homes and Health Telematics, Sherbrooke, Québec, Canada*, pages 75–82.

Kempke, B., Pannuto, P., and Dutta, P. (2015). Polypoint: Guiding indoor quadrotors with ultra-wideband localization. In *Proceedings of the 2nd International Workshop on Hot Topics in Wireless*, pages 16–20. ACM.

Martínez-Sala, A., Molina-Garcia-Pardo, J.-M., Egea-Ldpez, E., Vales-Alonso, J., Juan-Llacer, L., and García-Haro, J. (2005). An accurate radio channel model for wireless sensor networks simulation. *Communications and Networks, Journal of*, 7(4):401–407.

Ni, L. M., Liu, Y., Lau, Y. C., and Patil, A. P. (2004). LANDMARC: indoor location sensing using active RFID. *Wireless networks*, 10(6):701–710.

Röhrig, C. and Müller, M. (2009). Indoor location tracking in non-line-of-sight environments using a IEEE 802.15. 4a wireless network. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 552–557. IEEE.

Schantz, H. G. (2007). A real-time location system using near-field electromagnetic ranging. In *Antennas and Propagation Society International Symposium, 2007 IEEE*, pages 3792–3795. IEEE.

Texas Instruments (2014). CC1100 Single Chip Low Cost Low Power RF Transceiver. Document SWRS038D.

Venkatraman, S. and Caffery Jr, J. (2004). Hybrid TOA/AOA techniques for mobile location in non-line-of-sight environments. In *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, volume 1, pages 274–278. IEEE.