Efficient Indoor Proximity and Separation Detection for Location Fingerprinting

Mikkel Baun Kjærgaard
Department of Computer Science
University of Aarhus, Denmark
mikkelbk@daimi.au.dk

Georg Treu, Peter Ruppel, and Axel Küpper
Mobile and Distributed Systems Group, Institute for Informatics
University of Munich, Germany
{georg.treu|peter.ruppel|axel.kuepper}@ifi.lmu.de

ABSTRACT

Detecting proximity and separation among mobile targets is a basic mechanism for many location-based services (LBSs) and requires continuous positioning and tracking. However, realizing both mechanisms for indoor usage is still a major challenge. Positioning methods like GPS cannot be applied there, and for distance calculations the particular building topology has to be taken into account. To address these challenges, this paper presents a novel approach for indoor proximity and separation detection, which uses location fingerprinting for indoor positioning of targets and walking distances for modeling the respective building topology. The approach applies efficient strategies to reduce the number of messages transmitted between the mobile targets and a central location server, thus saving the targets' battery power, bandwidth, and other resources. The strategies are evaluated in terms of efficiency and application-level accuracy based on numerous emulations on experimental data.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Network Architecture and Design-Wireless communication

Keywords

LBS, Proximity/Separation Detection, Fingerprinting

1. INTRODUCTION

Location-based Services (LBSs) take into consideration the current positions of users or other targets in order to support navigation, to deliver a list of nearby points of interest like restaurants or to show buddies being in close proximity. LBSs can be realized in a *reactive* or *proactive* fashion. In the former category, location-based data is delivered to the user only on request, while proactive services are automatically triggered as soon as a pre-defined *location event* occurs, for example, when a target enters or leaves a city, district, building or another geographic zone. The user can then be informed about that event and receive additional information. Unlike reactive LBSs, proactive ones are much more difficult to realize, because targets need to be permanently tracked for checking the occurrence of location events. This paper focuses on two special problems that belong to the class of multi-target location events, where the positions of several targets need to be determined and compared on a permanent basis. *Proximity detection* is defined as the capability of an LBS to detect when two of a group of mobile targets approach each other closer than a pre-defined *proximity distance*. Analogously, *separation detection* discovers when two targets depart from each other by more than a predefined *separation distance*. The detection of such events can be used in manifold ways, for example, in the context of community or dating services for alerting the members of these communities when other members approach or depart. The solutions presented in this paper have been especially tailored for indoor environments like offices, factory floors, university campuses, hospitals, or railway stations.

In earlier work, mechanisms for proactive proximity and separation detection have been included into the LBS middleware TraX, see also [9] and [10]. These mechanisms control the positioning process within GPS-capable mobile devices carried by the targets and coordinate the transfer of the derived position fixes to a central location server for checking for proximity and separation with other targets. This transfer is referred to as *position updating*, and it may happen periodically, when the target has covered a certain distance with respect to the last reported position or if she has entered or left a certain zone. Proximity and separation checks are based on the *line-of-sight* or *Euclidean distance*, which can be simply calculated from the geographic positions of the involved targets. TraX applies a combination of different position updating and polling strategies with the goal to reduce the number of messages that pass the GPRS or UMTS air interface, to lower the battery consumption of the mobile phones, and to disburden the location server. Unfortunately, the use of GPS makes TraX applicable only in outdoor environments, because GPS signals typically do no penetrate buildings. Alternative outdoor positioning technologies, for

MOBILWARE 2008, February 13-15, Innsbruck, Austria

DOI 10.4108/ICST.MOBILWARE2008.2804

example cellular methods like Cell-Id, may work indoors, but lack in providing a sufficient degree of accuracy of position fixes as required for both detection schemes. Therefore, the only solution to offer proximity and separation detection within buildings is to use an indoor positioning scheme.

In the recent years, many indoor positioning schemes have been developed differing from each other in the kinds of signals used (infrared, radio, ultrasound), the type of signal measurements (signal traveling time, received signal strength, coverage) and the mathematical methods (fingerprinting, lateration, angle of arrival) for deriving a position fix from the measurements. One of the most prominent schemes is called *location fingerprinting* (LF). It estimates the position of a target from measuring the strength of radio beacons (*received signal strength*, RSS) emitted by several WLAN 802.11 access points in the close surrounding. The location of the target is then determined by mapping the measured values onto RSS patterns, which are called *fingerprints* and which have been pre-recorded at well-defined positions for storage in a map database. LF has been selected for extending the TraX framework, because it provides a comparatively high accuracy of location data when compared to other technologies. Another advantage is that it does not require dedicated hardware, that is, it works with existing WLAN 802.11 installations available in many buildings as well as with conventional WLAN-capable mobile devices.

Unfortunately, replacing GPS by LF in the TraX middleware is not enough. Unlike GPS, where mobile devices can determine their geographic position, LF only delivers a vector of RSS measurements as observed by the device on the spot. As a consequence, position updating cannot be triggered when the target has covered a certain distance or left a zone, but it requires a new position updating scheme, which carries RSS values and which is triggered by a certain change of RSS values. Another novelty concerns the semantic of distance. Checking for proximity and separation under consideration of Euclidean distances does not make much sense indoors, because several targets could be located on top of each other on different floors of a building, to give only one example. Applying both detection functions for walking distances is therefore a more reasonable, but also a more sophisticated approach.

This paper proposes different strategies for efficiently performing proactive proximity and separation detection in indoor environments based on walking distances and by using LF. Similar to its outdoor counterparts, the goal of these strategies is to lower the battery consumption of mobile WLAN devices carried by the targets, to reduce the workload of the server performing the checks and to keep the amount of messages passing the air interface as low as possible. The latter especially makes sense in cross-organizational scenarios, where position update and polling messages are not sent over the WLAN network used for performing LF, but by using public bearer services like GPRS or UMTS.

LF and advanced functions for LBSs have been a hot topic in research during the recent years. The following section gives an overview about related work and explains differences to and similarities with the approaches presented in this paper. Section 3 introduces the TraX middleware from a conceptual point of view and explains how to extend it for the purposes of indoor proximity and separation detection. Section 4 then describes position updating and polling strategies for both detection functions that work in combination with LF and walking distances. Finally, Section 5 presents the results achieved by prototype evaluation and emulation for the proposed strategies, followed by the conclusions and discussion of further work in Section 6.

2. RELATED WORK

In the recent years, LF has been evaluated and used mainly for single target location determination, therefore not addressing proximity and separation detection [2, 4, 12, 15], with NearMe [8] as an exception. NearMe supports a short-distance proximity detection, which takes into consideration RSS measurements and Euclidean distances only, as well as a long distance mode, which applies a base station coverage-graph analysis. NearMe is a client-server approach with periodic RSS updating between mobile device and location server, which causes significant overhead when a target does not move for a longer period of time.

LBSs applying LF in IEEE 802.11 networks and using proximity information have been built and evaluated for usability. The location-based messaging system InfoRadar [11], for example, uses an LF technique proposed by Roos et al. [12]. A location server polls RSS measurements from the targets' devices for estimating their positions and checking them for proximity subsequently. The ActiveCampus [14] system provides a set of LBSs to foster social-interactions in a campus setting. One of these services can list nearby buddies and show maps overlaid with information about buddies, sites and current activities. Targets are located using a terminal-assisted LF method proposed by Bhasker et al. [2] and a combination of poll-based and periodic RSS updating, which, however, turned out to be a bottleneck in this system when trying to scale beyond 300 concurrent users. The strategies proposed in this paper scale much better and are novel in that they consider walking instead of Euclidean distances, which, as mentioned before, better reflects the needs of indoor LBSs.

Several systems support the realization of LBSs based on LF in general. Many have been proposed for integrating position fixes produced by different positioning technologies, among them LF, thus easing implementation and improving server-side efficiency. Examples of such systems are the Rover system [13], the Location Stack [5] and its implementation in the Universal Location Framework (ULF) [3]. They provide means to integrate and fuse information from several positioning methods, query location information, improve scalability and define location-based triggers. The systems have been integrated with LF techniques applied in Horus [15] and RADAR [1]. Position fixes are obtained from the location sources by push, pull and periodic location updating methods. The Rover system has been evaluated for server-side efficiency in terms of CPU-load based on simulated inputs. In comparison, this paper proposes strategies for an efficient message transfer over the air interface, which also improves server-side efficiency and saves battery resources at the client-side.

3. TRAX

The strategies proposed in this paper for proximity and separation detection are part of the LBS middleware TraX [9], which has been developed for efficiently exchanging position fixes and for collecting, processing, and interrelating position fixes of several targets. The framework provides a

set of basic building blocks, which can be applied for a broad range of LBS applications and which can be dynamically configured, for example in order to meet accuracy and up-to-dateness demands on position fixes. The position management framework is arranged between a layer representing the on-target parts of one or several positioning methods and the LBS application, as illustrated in Figure 1. It is subdivided into so-called *low-level* and *high-level functions* and the on-server parts of positioning methods. The layer of the low-level functions sits on top of the on-target positioning methods and provides different methods for exchanging position fixes or position measurements between a mobile device and a location server. The high-level position management offers advanced functions for LBSs, for example proximity and separation detection as treated in this paper or k-nearest neighbor search and clustering. They apply the low-level functions according to a certain strategy. The on-server positioning methods sit in between the low-level and high-level layers and provide estimation of position fixes from position measurements.

TraX was originally tailored for outdoor use and for Euclidean-distance proximity and separation detection in conjunction with GPS, see the left of Figure 1. The low-level methods for exchanging position fixes include: position updating based on dynamically configuration of terminals for updating their positions when leaving a geographical update zone (*PU Zone*), and explicit *polling* of terminals for immediate reports of their positions (*PU Polling*). The high-level layer implements the functions of Euclidean-distance proximity and separation detection based on the so-called *Dynamic Centered Circles (DCC)* strategy [9].

In this paper, the middleware is extended for indoor use of walking-distance proximity and separation detection in conjunction with LF, see the right of Figure 1. The low-level methods for exchanging IEEE 802.11 RSS measurements include: RSS updating for sending RSS measurements when leaving a pre-configured update zone (*RSS-U Zone*), and explicit *polling* of terminals for immediate reports of RSS position measurements (*RSS-U Polling*). The high-level layer implements the functions of walking-distance proximity and separation detection based on the strategy proposed in Section 4.

LF positioning is supported in a *terminal-assisted* mode: the terminal conducts the RSS measurements and reports it to the location server, the latter usually on request or by sending periodic updates. The estimation of the target's location then happens at the server, which relieves the terminal from carrying the fingerprinting database and from applying complex estimation algorithms, thus enabling LF on resource-constrained terminals. In comparison, other LF architectures such as *network-based* or *terminal-based* setups can either not support resource-constrained devices or cannot be efficiently optimized in terms of message overhead as discussed in Kjærgaard et al. [7].

The RSS-U Zone method as presented in Kjærgaard et al. [7] is an RSS updating protocol that replaces the periodic updating of RSS measurements as usually practiced for terminal-assisted LF. Update zones are translated into compact RSS patterns, which can be passed to the terminal as a so-called *RSS detection request*. Based on its current RSS measurements and these patterns, the mobile device can decide whether it stays within or without the zone. Hence, RSS values are transmitted to the server only when

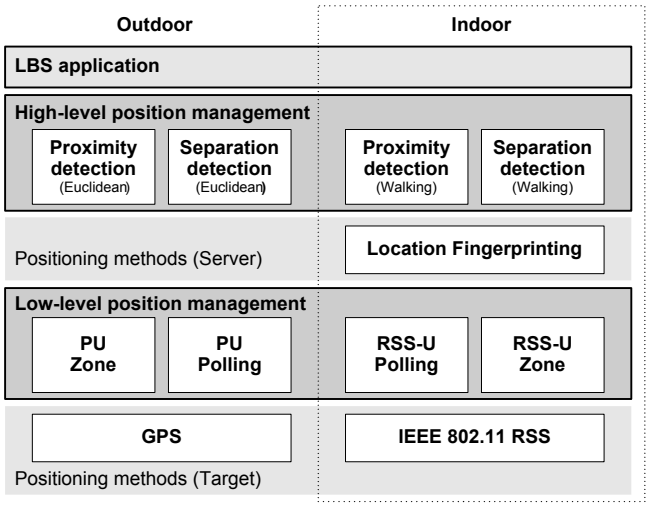

Figure 1: TraX

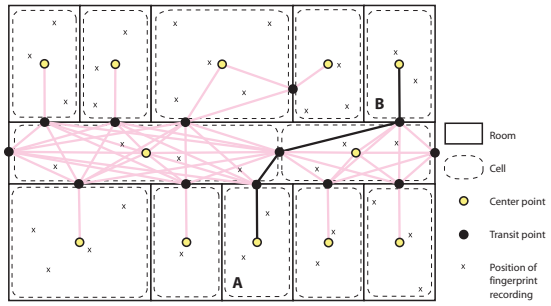

Figure 2: Walking distance between two cells.

needed and the overhead associated with periodic updating or polling is avoided. For deciding whether the terminal is within or without the zone with reasonable computational costs, a Bayes estimator is used that collapses the big probabilistic model over all locations available at the location server into a simpler one (maximum of 500 bytes), which distinguishes only between being within or without a configurable set of locations (the update zone). It turned out that this approach only induces little computational burden on the device and significantly saves the amount of messages passing the air interface when compared to periodic RSS updating. Despite of these advantages, it showed that the accuracy of the Bayes estimator is comparable to the classical approach.

4. APPROACH

The presented approach for indoor proximity and separation detection modifies the DCC strategy for working with walking distances and combines it with zone-based RSS reporting. The DCC strategy dynamically assigns each target update zones in order to correlate the positions of multiple targets. In indoor environments, such update zones can be effectively realized with zone-based RSS reporting, and walking distances between mobile users are much more relevant than Euclidean ones.

4.1 Walking Distances

For calculating walking distances, a topological building model must be constructed. A building can be described by a set of elements (rooms, corridors, stairways, etc.), all of which have a certain spatial expansion and one or more connection points to neighboring elements. A *cell* is defined as the basic unit of location the LF system can distinguish, that is, it is assumed that localization happens in terms of cells instead of coordinates. A cell usually covers small rooms or parts of a corridor. A more fine-grained discrimination is unrealistic, because of the moderate accuracy of current LF systems. Hence, building elements are always fully covered by one or more cells, and no cell can be part of more than one element. For simply calculating walking distances, the location of a target within a cell is always assumed to be the center point of the cell's enclosing rectangle. This model also solves the determination of walking distances between rooms on different floors.

However, a problem of this approach is that a target does not necessarily cross the center points of interjacent cells when walking from a source to a destination cell. To give an example, in Figure 2 cells on different sides of the corridor should be reachable directly and not by passing through the corridor cell's center point. As a solution, in addition to the center point, each cell is associated with a set of transit points, which connect a cell to neighboring cells. The topological model of a building is then defined as an undirected connected graph $B = \{P, E\}$, where P is the set of all center and transit points of all cells. The set of weighted edges E represents the distances between connected points. The center and transit points of one cell are always fully connected. Thus, the walking distance $d_{walk} : C \times C \rightarrow \mathbb{R}$ between two cells is defined as the length of the shortest path between their center points, which, however, may include passing interjacent cells through their transit points only.

4.2 DCC with Euclidian Distances

The classical DCC strategy includes a location server for monitoring the positions of several targets in order to detect when a pair of them gets closer to each other than a *proximity distance* d_p or when it separates by more than a *separation distance* d_s. The basic message flow between location server and device is as follows: when proximity or separation detection is requested for a pair of targets, their positions are first *polled* and compared. If the detection condition is already met, the requesting application is notified and the procedure stops. Otherwise, *position update requests*, which carry the definition of the update zones, are sent to both of the devices. The zones are chosen in a way that without any of the two devices triggering an update proximity and separation respectively cannot occur. The devices then continuously check generated position fixes against the update zone. In case of a match, a *position update* is sent to the location server. There, the reported position is compared to the update zones placed on the other target's device, which may or may not result in a need to poll it for its exact position as well. If, based on the exact positions, proximity or separation is detected, the application is notified and the procedure stops. Otherwise, new position update requests are sent to the devices.

The update zones in the DCC strategy are circle-shaped and centered around the terminal's last reported position.

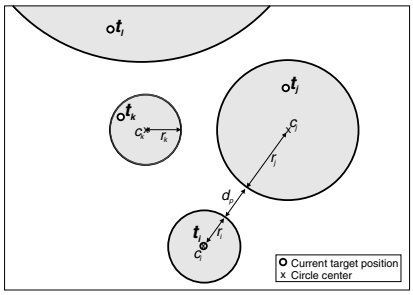

Figure 3: DCC with Euclidean distances.

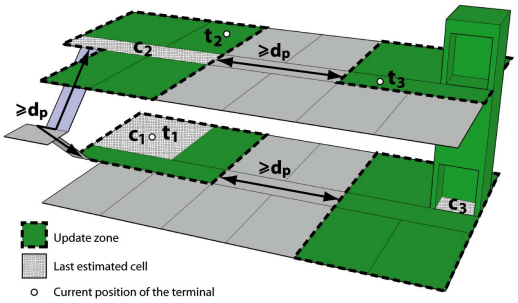

Figure 4: DCC for cells and walking distances

Positions are reported only when leaving the circle. For proximity detection, the circle computation works as follows, compare Figure 3: suppose t_i reports its current position and the neighbor of t_i with the closest circle turns out to be t_j. Assuming the circle of t_j has the radius r_j and the center point c_j, then t_i is assigned a new circle with center point c_i set to its current position and with radius $r_i := dist(c_j, c_i) - r_j - d_p$. In this way it is impossible that the distance between t_i and t_j can get below d_p without either of the two leaving its circle and reporting a position update.

For separation detection, suppose that from all targets t_j is farthest away from t_i, assuming that t_j is located at the border of its circle in opposite direction to t_i, which leads to the so-called maximum distance between both targets. The circle computed for t_i again has the center point c_i set to its current position, but the radius is set to $r_i := d_s - dist(c_j, c_i) - r_j$. Analogous to before, the distance between t_i and t_j can thus not exceed d_s without sending a position update. By choosing the neighbor t_j as described, the proximity and separation conditions are also guaranteed with respect to other possible neighbors t_i is tracked with.

4.3 DCC with Walking Distances

Indoor proximity detection based on walking distances uses the proximity distance $d_p > 0$ and an associated borderline tolerance $b >= 0$. Let c_i be the current cell of target t_i and c_j the cell of t_j. Furthermore, let $d_{walk}(c_i, c_j)$ be the walking distance between the targets' current cells as defined before. Then, proximity is checked by the following conditions:

1. If $d_{walk}(c_i, c_j) < d_p$, then proximity *must be* detected.
2. If $d_p \leq d_{walk}(c_i, c_j) \leq d_p + b$, then proximity *may be*

detected.

3. If $d_{walk}(c_i, c_j) > d_p + b$, then proximity *must not be* detected.

For separation detection based on the separation distance $d_s > 0$ the conditions are defined analogous. The purpose of the fuzziness interval given by the borderline tolerance b is to avoid excessive location reporting when the distance between t_i and t_j is approaching d_p. Without b, it would be necessary to track the devices on a very fine-grained level just to determine the exact moment when $d_{walk}(t_i, t_j)$ meets d_p. Put differently, the parameter b enables a trade off between desired detection accuracy and costs in terms of transmitted messages. In any way, it would not make sense to specify a higher detection accuracy than the accuracy of position fixes delivered by the used LF system. The reason for the gain in efficiency when using a bigger value for b is that, as described more extensively in [9], the minimum radius of the update circles used by the DCC strategy can be limited to $\frac{b}{2}$. Obviously, bigger circles lead to less position updates on average.

In order to apply the DCC strategy to the topological indoor model, the *walking distance space* (WDS) of a cell is introduced. Given a radius r, $WDS(c_i, r)$ of a cell c_i equals the set of all cells c_j whose walking distance $d_{walk}(c_i, c_j)$ to c_i is smaller than or equal to r. Hence, instead of geographical circle-shaped update zones centered around the last reported position, our adaption of DCC for indoors calculates the WDS with respect to a target's last estimated cell based on the calculated radius. This update zone, which is defined in terms of cells, is then configured at the targets' terminals by a respective *RSS detection request* using the RSS pattern technique described in [7]. The rest of the DCC algorithm basically remains the same: when a target t_i leaves its update zone, an *RSS update* is reported to the server. Based on the update, the current cell c_i of t_i is estimated. In case of proximity detection, the minimum walking distance m between c_i and the closest cell of the current update zones of all other targets t_j is calculated. If m is small enough so that proximity could occur, an *RSS polling* is issued to the respective target(s) t_j and its (their) current cell(s) c_j is (are) estimated as well. If, based on the cell estimates, the trigger condition is fulfilled, the application is notified. Otherwise, the minimum distance the targets t_i and t_j may walk without conflicting with one another, or with a zone of the other targets, is calculated. From these distances, two update zones (WDSs based on the estimated cells) are computed and assigned to the targets' terminals by means of new RSS detection requests. In case m was not too small before, only t_i is assigned a new update zone, reflecting a WDS with radius $r_i := m - d_p$. For separation detection the procedure is analogous.

As an example for proximity detection, Figure 4 shows a scenario inside a building, where the devices of three targets are configured with update zones (dark areas). Device t_1 has just reported an RSS update and its new update zone has been calculated as follows: the closest neighboring update zone to t_1's estimated cell was the one of t_3, so that the distance between the update zone assigned to t_1 and t_3 is as close to d_p as possible. As a consequence, the walking distance between the zone of t_1 and the zone of t_2 is larger than d_p (in the model distances along stairs are weighted heavier than horizontal ones).

5. EXPERIMENTAL RESULTS

For evaluating the approach, a simple location-based community service was implemented, which keeps the users of an office environment up-to-date about which persons of their buddy list are currently staying within a walking distance of p or smaller. Each possible pair of buddies is either observed for proximity or separation events. When a proximity event is detected, the buddy's name appears on the user's proximity list and separation detection is started for both of them. If, in turn, separation is detected, the person is removed from the list and proximity detection is restarted.

The fuzziness intervals for separation and proximity detection are made non-overlapping in order to avoid possible ping-pong effects. For a borderline tolerance of b, proximity detection is initialized with $d_p = p - b$ and separation detection with $d_s = p$. Thus, if the walking distance $d_{walk}(t_i, t_j)$ between two target persons t_i and t_j is below $p - b$, then they *must* appear on each other's proximity list. If $p - b \leq d_{walk}(t_i, t_j) \leq p + b$, then they *may* appear on the list. Finally, if $d_{walk}(t_i, t_j) > p + b$, then they *must not* be on the list.

5.1 Prototype

In order to show the practical feasibility of our approach with state-of-the-art equipment, a prototype was implemented and tested with Fujitsu Siemens Pocket LOOX 720 PDAs with built-in WiFi (IEEE 802.11) functionality. At the PDA, the functions for measuring RSS and evaluating RSS detection requests are implemented as a .NET application for Windows Mobile 2003 SE. The TraX server is implemented as a Java application, passing RSS detection requests to the PDAs and receiving RSS updates from the PDAs. Connectivity to the terminals was provided by a WiFi infrastructure using a proprietary protocol on top of TCP. For estimating locations from RSS updates and for computing RSS detection requests from sets of cells the TraX server utilizes an existing LF server.

A field test with two targets and an area spanning two floors with about 30 cells and 14 reachable base stations was conducted. After experimenting with different configurations, the proximity distance of the community service p was set to 12 m and the borderline tolerance b to 5 m. First, the targets walked in different patterns on the two floors. During one walk, a target went to the second floor while the other stayed on the first one. Then both targets walked to the second floor and back together. Finally, both walked up and back again, however, with the second target following at a certain distance.

From our experiences, it can be stated that the system worked properly and most of the time correct proximity and separation states were reported. However, also wrong or missing detections were experienced, which, apart from general LF inaccuracy, had two reasons: first, some communication delays happened as a result of roaming between the base stations used in the experiment. With the used combination of WiFi driver on the PDAs and type of WiFi access points, these delays amounted to several seconds, which made the system miss some detections and also report several detections in a bulk after the event had already passed. Second, the sampling rate of the used PDA is only 0.5 Hz, and hence the position derived at a device is delayed by up to 2 seconds. Considering both devices, the true distance between two targets then deviates from the measured one by up to 4

Figure 5: Walks recorded at two floors.

seconds of walking.

5.2 Emulation

In addition to the prototype and in order to obtain quantitative results, emulations were run based on data collected from a second test site. This test site offers 31 reachable WiFi base stations. It was divided up into 126 cells with an average size of 16 m^2 matching rooms or parts of hallways, spanning two floors. Each cell was fingerprinted by walking around in the cell for 60 seconds with a laptop that was equipped with an Orinoco Silver 802.11 card. After that, six sets of walks were collected, each comprising three 40-minutes-walks simultaneously performed with three devices, totaling about 12 hours. The fingerprinting and walk collection were separated by several weeks. Three of the six walk sets were recorded by the PDAs also used for the prototype. The other three used the laptops with the Orinoco cards. The RSS values were collected at a sampling rate of 0.5 Hz and 1 Hz respectively. Each sample of a walk contains a time-stamp, the measured RSS values of the surrounding base stations, as well as the current ground truth, which was manually specified on a laptop-shown map. During the recording of a set of walks always one of the three devices was kept stationary, while the other two were carried along different routes through the building. The targets walked at moderate speeds, with several pauses and over two alternating floor levels, compare Figure 5.

Based on the recorded data the approach was examined in terms of efficiency and accuracy. For that, from the zone detection methods presented in [7] the Bayes estimator was selected. As a benchmark for comparison, a *reference strategy* based on terminal-assisted LF with periodic RSS reporting at 1 Hz was assumed. In this way, for all possible pairs of targets and at every moment in time the location server can decide whether the proximity criterion is met or not. For location estimation from reported RSS values at the server-side the same LF system, which is based on the techniques described in [4], was used by the proposed DCC strategy as well as by the reference strategy. The PDA's RSS measurements were normalized to match the fingerprints collected with the Orinoco cards using the method proposed in [6].

As explained before, three operations are needed for target tracking: RSS detection requests, RSS updates, and RSS pollings. While DCC combines all three operations, the reference strategy only uses RSS updates. Each of these operations causes one message in the uplink and another one in the downlink. The only exception are RSS updates in the DCC strategy. They need no explicit acknowledgement in the downlink, because they are always confirmed by a new position RSS update request message. Technically, up- and downlink have different resource-consuming properties and should be treated separately. For brevity, however, they are not distinguished in the following and the total number of messages transferred per target is summed up.

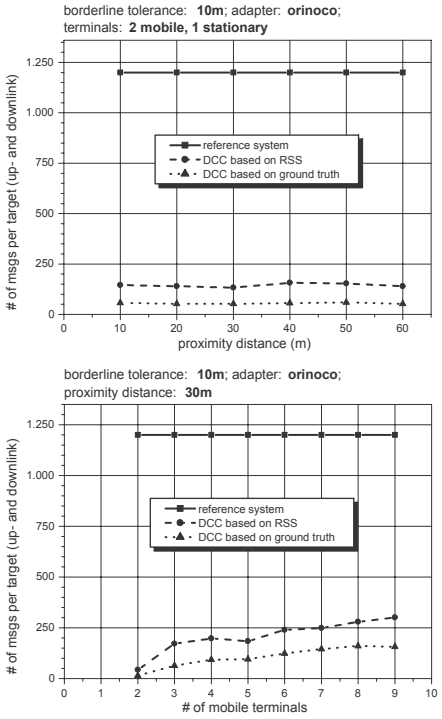

Figure 6: (a) # of messages dependent on proximity distance p, (b) # of messages dependent on number of terminals

Another issue is the amount of transferred data. While message acknowledgments as well as polling requests (the downlink message of an RSS polling) are very lightweight, RSS updates as well as polling responses carry measured RSS values, which amounts to more data. For example, the Orinoco and the PDA walks contain on average around 5-7 base stations per sample. Furthermore, experiments with an Apple Airport Express card yielded about 14 visible stations at a time. However, in practice only the 5-7 strongest stations need to be reported, because including more stations will not significantly increase the accuracy. Thus, the size of an RSS update has an upper limit, which, however, is dependent on the underlying technology. The RSS detection request messages (downlink) have the biggest size, which, according to [7], can be limited to 500 bytes for the Bayes estimator. For the other (more inaccurate) RSS detection methods, the size is typically smaller.

Whether the goal is to save transferred bytes or messages depends on the constraints considered. Monetary costs for transmission over public bearer services like GPRS or UMTS are typically billed according to data volume in bytes. On the other hand, server scalability is rather constricted by the number of messages that have to be handled in the uplink. Considering physically limited resources like the air-interface

or the battery power at the device used for message sending and receiving, the number of transmitted frames seems most critical. For IEEE 802.11 this figure equals the number of transferred messages, because all described message types are small enough to fit within one 802.11 frame. Therefore and also because the number of bytes per message can be specified rather arbitrarily, the following evaluation only discusses the number of transferred messages.

For evaluating message efficiency, three parameters were varied: the proximity distance p, the number of terminals observed in a pairwise fashion (i.e., the size of the buddy list), and the borderline tolerance b. Additionally to the DCC and the reference strategy based on collected RSS values, DCC was also performed on ground truth, which behaves as if the RSS detection requests worked with perfect accuracy.

Figure 6a shows the number of messages transferred per target dependent on p averaged for the three walk sets collected with the Orinoco cards. The time was normalized to 10 minutes. Three things become apparent: first, in comparison to the reference strategy, DCC based on RSS reduces the amount of messages strongly (about factor 9). Second, the performance of all three approaches is rather independent from the chosen proximity distance. While this was expected for the reference strategy, which steadily sends 120 messages per minute, for DCC this can be explained by the fact that independent of the current distance of a pair of targets and p, both of them are permanently observed either for proximity or for separation events. The third observation is the difference between the performance of DCC based on RSS and DCC based on ground truth. The former triggers about 2.5 times as much messages as the latter. Obviously, the employed RSS detector (Bayes estimator) triggers a number of wrongly sent RSS updates, which do still belong to the cells contributing to the update zone and which are therefore correctly not sent by DCC based on ground truth. However, it can be stated that the difference between the real and the ideal DCC detector is still acceptable when taking into account the savings compared to the reference strategy. Also, it must be stated that the collected walks represent a mobility pattern presumably more mobile than in a typical office scenario.

Figure 6b shows the number of messages per target dependent on the number of pairwise observed targets. For this, all of the $3 * 3 = 9$ walks collected with the Orinoco cards were aligned in time and played simultaneously. Expectedly, the number of messages per target used by the reference strategy stays the same, while for DCC it increases. The proportion between messages sent by DCC based on RSS and DCC based on ground truth starts with a value of 2.8:1 for two targets, then slowly decreases with an increasing number of targets and settles at a value of about 1.8:1 for five to nine targets. The slope of the DCC curves is not too steep, so that the approach seems practicable even for bigger buddy lists. Note that the number of targets tracked pairwise (equals the size of the buddy list) is not equal to the number of users of the community services. While our aim is to make the service scalable to thousands of users, this examination was related to the size of a single user's buddy list, that is, the number of users she constantly wants to keep track of, a figure which is assumed to be rather small. Thus, by limiting the number of messages per user as described before, server scalability in terms of the number of users is improved.

Figure 7a depicts the message overhead dependent on the borderline tolerance b. For the Orinoco cards as well as for the PDAs, all three-person-walk sets were averaged. Two observations are noteworthy here: first, the number of messages in all configurations decreases by roughly the same factor of about 50 % from $b = 1$ to $b = 24$. This can be explained by taking into account that the minimum radius measured in walking distance of a DCC zone is limited to $\frac{b}{2}$. Thus, with an increasing b the minimum zone size increases, which leads to a decreasing number of RSS updates. The second observation is that DCC with RSS performs considerably worse for the PDAs than for the Orinoco cards (the factor ranges between 2.6 and 3.8). One reason for this may be that the PDA's RSS measurements need to be normalized as described before to match the fingerprints in the database, which were collected with the Orinoco card. The normalization function does, however, not perfectly account for the difference in RSS measuring between the Orinoco card and the PDA, which degrades accuracy in general. Hence, the RSS detectors at the PDAs produce more wrongly sent RSS updates.

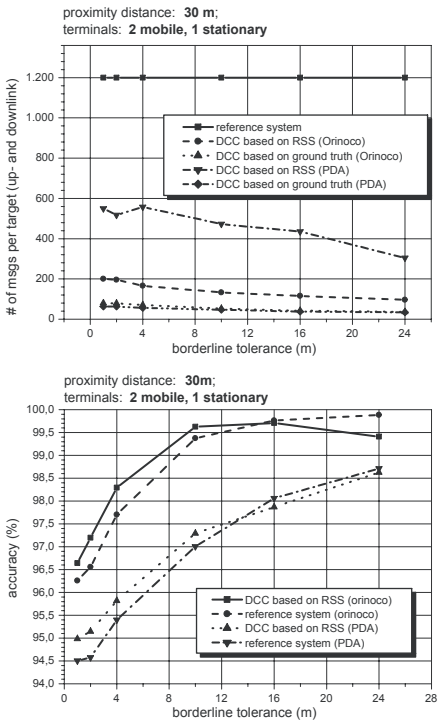

Figure 7: (a) # of messages dependent on borderline tolerance b, (b) Accuracy dependent on borderline tolerance b

The application-level accuracy of the presented strategies is analyzed according to a simple metric: based on the ground truth at each moment in time and for each pair of tracked targets t_i and t_j, the current walking distance $dist(t_i, t_j)$ is computed. It is mapped onto a state $X \in \{P, F, S\}$ with $X = P$ if $dist(t_i, t_j) < p - b$ (t_i and t_j are in proximity), $X = F$ if $p - b \leq dist(t_i, t_j) \leq p + b$ (they are within the fuzziness interval), or $X = S$ if $dist(t_i, t_j) > p + b$ (they are separated). Based on this mapping, the number

of situations (time frames of one second) are counted where the DCC and the reference strategy indicate a wrong state information, that is, when the state X_{DCC} or X_{ref} deviates from the ground truth X_{gt}. However, a wrong state information is only logged when $X_{gt} = P$ or $X_{gt} = S$, because within the fuzziness interval both states are allowed. The metric is very simple, because in the tested service there is an interplay between proximity and separation detection. For testing the events separately, it would be necessary to consider false and true positives and negatives respectively and derive from that metrics like sensitivity and precision. In this case, however, a positive with respect to proximity detection is a negative for separation detection. Since both situations ($X = P$ and $X = S$) have a comparable probability (dependent on the building layout and the proximity distance), the two event types actually cancel each other out and hence one accuracy metric suffices.

Figure 7b plots the achieved accuracy (that is, the percentage of situations where no wrong state information is given) for the DCC as well as for the reference strategy. First, for all curves the accuracy increases with an increasing borderline tolerance, which is due to the decreasing impact of LF inaccuracy on distinguishing the states S and P. Second and confirmatory for the good applicability of the DCC strategy, its accuracy is generally not worse than that of the reference strategy. It performs even slightly better for a low borderline tolerance and slightly worse for higher borderline values. Third, the Orinoco measurements yield a higher accuracy than those of the PDAs. However, it can be stated that in general a high accuracy is achieved (all four strategies are always above 94.5 %), even for a low borderline tolerance.

6. CONCLUSION AND FURTHER WORK

The paper has demonstrated that proactive proximity and separation detection can be effectively realized for indoor environments, while being resource-aware at the same time. The evaluation showed that the presented approach can decrease the number of transmitted messages with a factor of 9. The approach is feasible for very resource-limited devices like mobile phones or active tags and makes use of state-of-the-art LF technology and device hardware. Also, despite the general inaccuracy of LF, it turned out that at an application level a rather high detection accuracy above 94.5% can be achieved. A possible extension to the described community service, which recognizes targets closer than a static threshold, would be a buddy tracker that constantly shows the user a sorted list of the n-nearest-neighbors among his buddies. One piece of future work is to show how such a service can be realized by dynamically applying proximity and separation detection to pairs of targets.

Acknowledgements

M. B. Kjærgaard is partially funded by the software part of the ISIS Katrinebjerg competency centre.

7. REFERENCES

[1] P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proc. of IEEE INFOCOM*, 2000.

[2] E. S. Bhasker, S. W. Brown, and W. G. Griswold. Employing user feedback for fast, accurate, low-maintenance geolocationing. In *Proc. of the 2nd IEEE Int'l Conf. on Pervasive Computing and Communications*, 2004.

[3] D. Graumann, W. Lara, J. Hightower, and G. Borriello. Real-world implementation of the location stack: the universal location framework. In *Proc. of the Fifth IEEE Workshop on Mobile Computing Systems and Applications*, 2003.

[4] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proc. of the 10th Int'l Conf. on Mobile Computing and Networking*, 2004.

[5] J. Hightower, B. Brumitt, and G. Borriello. The location stack: a layered model for location in ubiquitous computing. In *Proc. of the 4th IEEE Workshop on Mobile Computing Systems and Applications*, 2002.

[6] M. B. Kjærgaard. Automatic mitigation of sensor variations for signal strength based location systems. In *Proc. of the 2nd Int'l Workshop on Location- and Context-Awareness*, 2006.

[7] M. B. Kjærgaard, G. Treu, and C. Linnhoff-Popien. Zone-based RSS reporting for location fingerprinting. In *Proc. of the 5th Int'l Conf. on Pervasive Computing*, May 2007.

[8] J. Krumm and K. Hinckley. The NearMe wireless proximity server. In *Proc. of the 6th Int'l Conf. on Ubiquitous Computing*, 2004.

[9] A. Küpper and G. Treu. Efficient proximity and separation detection among mobile targets for supporting location-based community services. *ACM SIGMOBILE Mobile Computing and Communications Review*, 10(3):1–12, July 2006.

[10] A. Küpper, G. Treu, and C. Linnhoff-Popien. TraX: A device-centric middleware framework for location-based services. *IEEE Communications Magazine*, 44(9):114–120, September 2006.

[11] M. Rantanen, A. Oulasvirta, J. Blom, S. Tiitta, and M. Mäntylä. InfoRadar: group and public messaging in the mobile context. In *Proc. of the Third Nordic Conf. on Human-Computer Interaction*, 2004.

[12] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen. A probabilistic approach to WLAN user location estimation. *Int'l Journal of Wireless Information Networks*, 9(3):155–164, 2002.

[13] S. Banerjee et al. Rover: Scalable location-aware computing. *Computer*, 35(10):46–53, October 2002.

[14] W. G. Griswold et al. ActiveCampus: experiments in community-oriented ubiquitous computing. *Computer*, 37(10):73–81, 2004.

[15] M. Youssef, A. Agrawala, and U. A. Shankar. WLAN location determination via clustering and probability distributions. In *Proc. of the First Int'l Conf. on Pervasive Computing and Communications*, 2003.