Self-Mapping in 802.11 Location Systems

Anthony LaMarca, Jeff Hightower, Ian Smith, Sunny Consolvo

Intel Research Seattle

Abstract Location systems that are based on scanning for nearby radio sources can estimate the position of a mobile device with reasonable accuracy and high coverage. These systems require a calibration step in which a map is built from radio-readings taken on a location-aware device. War driving, for example, calibrates the positions of WiFi access points using a GPS-equipped laptop. In this paper we introduce an algorithm for *self-mapping* that minimizes or even eliminates explicit calibration by allowing the location system to build this radio map as the system is used. Using nearly 100 days of trace data, we evaluate self-mapping's accuracy when the map is seeded by three realistic data sources: public war-driving databases, WiFi hotspot finders, and sporadic GPS connectivity. On average, accuracy and coverage are shown to be comparable to those achieved with an explicit war-driven radio map.

1 Introduction

Location has long been established as a key element of a mobile device's context [13]. One class of location systems, which we refer to as *beacon-based* location systems, estimates the position of a device by referencing nearby radio sources. Beacon-based systems have been shown to provide both good accuracy and high coverage in a variety of settings [7, 10]. Creating the necessary map of the radio sources, however, can be a costly and time-consuming process. In this paper, we introduce and evaluate an algorithm for *self-mapping* that enables beacon-based systems to build a radio map on-the-fly, while the system is in use.

Beacon-based systems typically use signals from 802.11 access points (APs) [1, 4], but Bluetooth devices, GSM cell towers, FM radio transmitters and combinations of these have also been used [7, 10]. Beacon-based systems have two advantages over other types of location systems. First, the huge number of fixed radio sources in the environment provides high coverage [10]. Second, because beacon-based systems use standard radios built into the user's device, there is no need for special hardware.

A drawback of beacon-based systems that estimate a device's absolute location is that they require a map of nearby radio sources to interpret the radio-scan results. For systems like Active Campus [4] that employ a predictive radio model, the map takes the form of the latitude and longitude of each 802.11 access point. In the case of "fingerprinting" systems like RADAR [1], this map is composed of a collection of radio-scan results and the locations in which those scans were taken. In all cases, these systems depend on this data and cannot translate radio observations into location estimates without it.

Typically, the radio map is manually built for the areas of interest, loaded onto the appropriate clients or servers, and then deployed to users. Re-mapping may be

performed occasionally to take into account new radio beacons that have been deployed or old ones that have been moved or decommissioned. Both the initial mapping and subsequent re-mappings are time consuming and represent the biggest cost in deploying and maintaining the system. In this paper, we propose a *self-mapping* algorithm that can reduce or even eliminate the overhead of mapping.

With self mapping, only a small amount of initial map data, what we call *seed data*, is needed to deploy the system. As the location system runs, radio scans are used to "grow" the seed data into a full radio map, as well as estimate the device's location. An immediate benefit of this approach is the time and energy saved by system managers, as they do not need to collect a full set of map data to deploy their system. Another advantage is that the system managers need not predict exactly which geographic areas must be mapped for the system to be useful, as the map will be refined in the most heavily used areas. Self-mapping also obviates the need to manually refresh the radio map. New radio sources will automatically and incrementally be incorporated as the system is used. The challenge in building a self-mapping system is that with a small set of mapping data, the radio scans are of limited use in meaningfully extending the map.

We present a graph-based algorithm for mapping radio beacons given a small set of seed data:

- Nodes in the graph represent radio beacons, and weighted edges represent the nearness of the two beacons.
- Edge weights are determined by using the radio scans to infer which beacons are near which other beacons.
- By anchoring the beacons in the seed set at their known locations, we can treat the graph as a constraint problem and compute likely locations for the unknown beacons by minimizing constraint violations.

To test our self-mapping algorithm, we gathered radio trace data from three volunteers over an eight-week period, totaling nearly 100 days worth of traces containing over 20 million radio readings. Using this data, we evaluated how well the maps produced by our algorithm compared both in accuracy and coverage to a traditional radio map when used by a location system. For the location system, we used the 802.11 scanner and the centroid tracker from the Place Lab toolkit [10]. We tested our algorithm with seed data from a variety of real-world sources, including public war-driving databases, WiFi hotspot finders, and sporadic GPS connectivity.

Our results show that self-mapping works well in practice and is a viable alternative to explicit mapping. Due to the large amount of structure that can be extracted from radio traces, self-mapping with a seed set containing as little as 10% of the radio sources can build a radio map that provides almost 90% coverage. Self-mapping with a seed set drawn from a public war-driving database provided median accuracy of 80 meters with 96% coverage while a seed set built from the user's own sporadic GPS coverage (so-called "opportunistic war driving") provided median accuracy of 56 meters with 84% coverage.

Our results also show that for an up-to-date radio map, self-mapping can add new beacons with nearly the same accuracy as war-driving. We show that our self-mapping algorithm's beacon location estimates were within 31m of an AP's true location on average, 5m worse than a war-driving estimate.

This paper is presented in three parts. First we outline how radio maps are currently built and describe our self-mapping technique. Then, we explain our data-collection strategy and sources of seed data. Finally, we present our results and conclusions.

2 Related Work

In this section, we describe how radio maps are built for two types of beacon-based location systems: those using predictive radio models and those using radio fingerprinting. We also discuss how self-mapping has been applied in symbolic location systems as well as sensor networks and robotics.

Estimating Location with a Predictive Radio Model

Both Active Campus and Place Lab are beacon-based systems that use a predictive radio model to estimate location [4, 10]. Radio models range from simple heuristics (*e.g.,*. 802.11 can be heard for 100 meters in all directions from the access point) to complex ones that consider antenna types and the attenuation of various building materials [12]. The radio map in these systems contains an estimated position of each known radio beacon as well as other possibly useful characteristics (*e.g.,* antenna height or transmit-strength). This data can be directly measured, as was done in the Active Campus deployment during which the locations of approximately 200 802.11 APs were manually determined.

Less accurate, but faster and not requiring physical access, beacon locations can be inferred by measuring where they can be observed. This technique is often performed by the "war driving" community to find the location of 802.11 APs. War driving requires a laptop or PDA with 802.11, GPS, and one of many publicly-available war-driving programs (netstumbler.com, kismetwireless.net). These programs log the APs that they hear, and estimate the beacons' locations using a predictive model. Our results in Section 6.3 show that for a small sample of APs, war-driving estimates are within 26m of the true location on average. War driving data has been shown to be an effective source of radio map data, allowing a user to be located using 802.11 with 15-20m accuracy in areas with high AP density [10].

The simple nature of the mapping data used by predictive systems like Active Campus makes them good candidates for self-mapping. In addition, 802.11 AP locations can be downloaded from public sources like WiFi clubs and war-driving databases. For these reasons, we chose to develop our initial self-mapping algorithm for a predictive beacon-based location system.

Estimating Location with Radio Fingerprints

Fingerprinting-based location systems are powerful and simple. In a fingerprinting system, the radio map takes the form of radio-scans tagged with the location where the scans were performed. When a device wants to estimate its location, it performs a scan, finds the radio scan in the map that it most closely matches, and estimates the device to be located where that matching scan was taken. To reduce errors, these systems generally average the location of the closest k scans where k is a small number, often 4.

A benefit of the fingerprinting approach is that there is no need to model radio propagation; it simply remembers what should be seen and where. RADAR [1] and Ekahau (ekahau.com) are fingerprinting systems that are accurate to 1-3m. Unfortunately, to perform this well, these systems require a highly-detailed and densely populated radio map compared to predictive systems. For example, the RADAR experiments employed a radio map with approximately one scan every square meter.

The match between fingerprinting and self-mapping is not as clear as for predictive systems. While high coverage can likely be obtained, accuracy may degrade, undermining the key advantage of fingerprinting. Second, all of the public radio-beacon databases we have found exported only summaries and none provided the raw radio scans that would be needed to seed a fingerprint radio map. Thus, finding a workable seed set for self-mapping with fingerprinting presents a challenge.

Self-Mapping in Sensor Networks

A variety of self-mapping algorithms have been developed for sensor networks. Some assume that a small percentage of sensor nodes know their own location and the algorithms estimate every sensor node's absolute location. Other algorithms assume no information and estimate only the relative location of sensor nodes. Some sensor network self-mapping algorithms use only the knowledge of which other nodes are in range [6], while others utilize observed signal strength [2], or acoustic ranging [3]. The graph structure used by many of these algorithms is similar to what we present in Section 3. The key difference is that in the case of sensor networks, the sensor nodes are trying to locate themselves, and can directly measure which other nodes are in range. In our case, we observe radio beacons from an independent vantage point and from these observations, must indirectly infer the relationship between beacons.

Priyantha et al. expand on the algorithms used in fixed sensor networks to show how mobile devices can aid non-mobile devices in estimating their location [11]. They show that with assistance from mobile device, fixed devices can locate themselves more efficiently and can also disambiguate so called "non-rigid" configurations. This work is similar to ours in that a mobile device aids in the location estimation of a collection of non-mobile devices. In their work, location is computed using a mix of readings from mobile and fixed devices, while ours is more extreme and only employs measurements taken by mobile devices. Priyantha et al. employ a polynomial-time closed-form solution to find the optimal assignment of locations, an acceptable approach for the small 16-256 node networks they consider. Given that hundreds of thousands of radio beacons may be in a metropolitan area, we employ a less accurate, but faster iterative approach, to approximate the optimal location assignments given the readings.

Other Approaches

Of all the 802.11-based location systems, NearMe comes the closest to performing self-mapping [8]. NearMe is a service designed to determine when two devices are in proximity. As an extension, NearMe allows 802.11 APs to be associated with physical places (*e.g.,* '2nd floor copy room') or resources (*e.g.,* 'duplex printer'). NearMe uses radio traces to build a neighborhood graph of which APs are near each

another out to eight hops. Although NearMe does not estimate absolute locations, it does estimate ranges between APs based on the traces and the neighborhood graph.

Laasonen et al. [9] built a cell-phone application that constructs a graph of a user's "places" based on the cell towers he visits with his phone. To overcome the limitation of knowing only the single cell tower the phone is currently associated with, the graph is built by observing transitions between cells.

Self-Mapping in Robotics

Robots can construct accurate maps of their environment during normal operation using a technique known as Simultaneous Localization and Mapping (SLAM). SLAM is similar to the work in this paper, however much of the research in SLAM focuses on solving the data-association problem. Data association is the challenge of matching environmental features that the robot observes (typically using a camera or laser range-finders) with the contours or features in the partial map [5]. Our self-mapping task has no data association problem because access points and other beacons all have unique IDs. When a device sees an ID in a scan there is no ambiguity about which beacon it is. Robots also typically make heavy use of odometry or inertial sensors which are uncommon on mobile and handheld devices.

3 A Graph Algorithm for Self-Mapping

In this section we introduce our algorithm for self-mapping the locations of radio beacons. For clarity, we describe our algorithm in the context of 802.11. The same algorithm should work with other radio technologies, provided the ID and observed signal strength are available from the radio scans. For simplicity, we present examples using a two-dimensional representation of location, but this algorithm also works in three dimensions.

3.1 Inputs

We assume that there are two sets of data available: The *seed set* and the *radio traces*. The seed set contains the unique identifier *(id)* and location *(latitude, longitude)* of a set of beacons with known location.

$$SeedSet = \{(id_1, lat_1, lon_1) , (id_2, lat_2, lon_2) , \ldots , (id_n, lat_n, lon_n) \}$$

The radio traces are made up of a collection of time-stamped radio scans. Each radio scan contains a timestamp *(ts)*, zero or more beacon identifiers *(id)*, and the observed signal strength *(ss)*.

$$RadioTraces = \{RadioScan_1, RadioScan_2, \ldots , RadioScan_m \}$$

$$RadioScan = (ts, \{(id_1 , ss_1) , (id_2 , ss_2) , \ldots , (id_o , ss_o) \})$$

3.2 Exploiting Signal Strength

If a radio scan sees more than one beacon, we know that since both beacons were observed from one location, they must be within twice the maximum transmit radius of each other. In the case of 802.11, this means that the two APs are likely within 400-500m of each other. We can often tighten this rough constraint by using the signal strength with which each beacon was observed. Since the falloff in signal strength is weakly correlated with an increase in distance [12], we can use it as a hint that the user was either close or far from each of the beacons. Signal strength does not tell us the angle between the beacons, but it still helps constrain the maximum distance. Consider the scans depicted in Figure 1. In the first scan, beacons b_1 and b_2 are both observed with weak signal strengths, establishing a distant proximity. The second scan shows a stronger reading (*-60dBm*) for b_1. This provides new information: since b_1 was heard strongly, the user was probably close to b_1, and since b_2 was also heard from the same location, b_1 and b_2 are probably closer than twice their maximum range. The third reading makes the likely separation between b_1 and b_2 smaller still, as both are observed with a reasonably strong signal.

To quantify this distance, we use Seidel's model for propagation of signals in the wireless networking band [12]. According to the model, expected signal strength (*ss*) at distance *d* is:

$$ss = ss_0 - 10 \cdot n \cdot \log_{10}(d/d_0)$$

where ss_0 is the signal strength that would be observed in free space at reference distance d_0 from the transmitter. The constant *n* is based on characteristics of the particular radio and the physical environment (density of obstacles, etc), with *n* typically varying between 2 and 5. At 1m, 802.11 APs do not typically return signal strength greater than -32dBm, so we use $d_0 = 1$ and $ss_0 = -32dBm$. Since we operated in a city with primarily wood and glass structures, and since we can assume nothing about the radios being modeled, we chose *n* to be on the low side: $n = 2.5$. Solving for *d* with these values, we get:

$$ss = -32 - 10 \cdot 2.5 \cdot \log_{10}(d/1)$$

$$d = 10^{(-32-ss)/25}$$

This establishes an estimated distance between the user and the observed beacons.

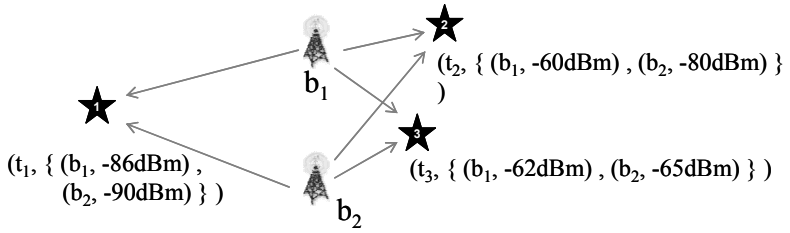

Fig. 1. An illustration of three scans that see the same two beacons. Starting with scan 1 on the left and proceeding clockwise, each scan reveals the two beacons to be closer together as their combined signal strengths grow larger

Given a pair of observations for two different beacons, we can establish the following constraint on the two beacons.

$$\textit{The distance between } b_1 \textit{ and } b_2 \leq 10^{(-32-ss1)/25} + 10^{(-32-ss2)/25} \quad (1)$$

In the example depicted in Figure 1, we only compared beacons that were observed in the same scan. In reality, because scans are taken at a rate of 1-4Hz, the user is unlikely to move far during a scan. Thus it is reasonable to compare across a few scans taken in rapid succession. If beacon b_i is observed in one scan, and beacon b_j is observed in a successive scan 500ms later, it is fairly safe to assume that the scans were taken "in the same place" and to infer a proximity relationship between b_i and b_j based on these scans. Varying the acceptable time threshold trades off the accuracy of the constraints against the connectivity of the graph: increasing the threshold allows more edges to be added to the graph, but also increases the error introduced by the user's motion[1]. To balance these factors, we ran our experiments using a time threshold of 1 second.

3.3 Graph Construction

The graph contains one node for every beacon that appears in either the seed set or in a scan from the radio traces. The radio traces are then traversed in time order and a fixed sliding-window's worth of scans are considered. (We used a one second window in our experiments). For each window, we select the strongest observation of each beacon seen in the window, and then consider all pairings of these beacons. For each of these pairs of beacons b_i and b_j, we estimate the maximum distance d_{max} between b_i and b_j using Equation 1 and the strongest signal strengths ss_i and ss_j. If no edge exists in the graph between the node for b_i and b_j, one is added with weight d_{max}. If there is an existing edge with a higher weight, it is replaced with an edge with weight d_{max}.

When all of the radio traces have been considered, the locations of the beacons in the seed-set are assigned to their nodes. Since these nodes already have a known location, we call them *anchor nodes*. The remaining task is to assign locations to the non-anchor nodes while trying to minimize violations of the constraint that for all nodes i and j that are connected by an edge, the distance between them should be less than the weight of that edge.

3.4 Finding a Solution

We assign locations to nodes using the following iterative approach. First, we assign all non-anchor nodes the location of the nearest anchor node in the graph. (If no path exists between a node and an anchor node, the node will not be assigned a location.) This location assignment will almost certainly violate a large number of edge constraints which we correct by iteratively reducing the error in the graph.

---

[1] An improvement to a simple time threshold would be to allow edges to be added for beacons seen many seconds apart, factoring the likely distance moved into the edge weight

The error of an edge is defined to be the proportion by which the edge's nodes are violating their shared constraint. If two nodes are 80m apart and share an edge of weight 100, the edge has zero error. If two nodes are 250m apart and share an edge of weight 100, the edge has an error of (250/100 – 1.0) or 1.5. The error for a node is defined to be the sum of the error of all of its edges. The error of the graph is defined to be the sum of the error of all nodes.

Error is reduced by repeatedly choosing a random non-anchor node and trying to reduce its error. We attempt to reduce a node's error by considering random Gaussian-distributed alternate locations for this node and measuring how they would affect the node's error. If a reduction in error can be achieved, the node is moved and the errors for that node and its neighbors are updated. This continues until either error for the graph is reduced to zero (meaning a solution satisfying all constraints has been found) or improvements have reduced to an unproductive level. When the algorithm completes, the locations of the anchor and non-anchor nodes are exported as a new radio map.

4 Test Data

Because self-mapping occurs as users go about their daily lives (*i.e.,* the areas in which trace data is collected is unpredictable), it would be unrealistic to test our algorithm using war driving radio traces that make systematic, serpentine traversals of a collection of streets. Therefore, we collected radio-traces from three volunteers from Intel Research Seattle as they went about their daily lives. Volunteer 1 (V1) lived 1.5 miles east of our lab and commuted to and from work by walking, Volunteer 2 (V2) lived 1 mile west and commuted by a mix of walking and driving, and Volunteer 3 (V3) lived 3 miles south and commuted by bus.

To collect the trace data, the volunteers carried laptops outfitted with hardware and software to continuously collect 802.11 and GPS traces; external batteries were provided so that the volunteers would not need to recharge the laptops throughout the day. Except for having to carry the laptop at all times, volunteers were asked to behave as they normally would. The one exception to this request was that we asked the volunteers to place their laptop near a window when they were in a vehicle to maximize the chance of obtaining a GPS lock.

The trace data was collected over a period spanning the end of 2004 and the beginning of 2005 (each volunteer collected data for four to eight weeks). During this period, we collected a total of 98 days worth of radio traces, consisting of 13.3 million 802.11 readings and 7 million GPS readings. A total of 9450 unique 802.11 APs were seen in the logs. Overall, there was a GPS lock 20.5% of the time and at least one 802.11 AP was seen in 90.2% of the scans.

Using these logs and a variety of self-mapping scenarios described in the next section, we tested both the coverage and accuracy offered by the radio maps generated by our self-mapping algorithm.

5 Self-Mapping Scenarios

One of our goals was to investigate the effectiveness of self-mapping in real-world scenarios. Accordingly, we identified three real-world sources of seed data to bootstrap self-mapping. This section describes each of these sources, as well as the war-driving scenario we use as the control in our experiments.

5.1 Self-Mapping with Sporadic GPS Coverage

As GPS chipsets become smaller and cheaper, more mobile devices will include integrated GPS. Since GPS usually requires a clear view of the sky, it seldom works indoors and does not provide sufficient coverage to be used as the sole source of location information. It can, however, be used as a source of seed data for self mapping. In this scenario, the device starts out with an empty seed-set. At the end of each day, that day's traces are mined for any trace fragments in which the user had a GPS lock. These log fragments are treated as a set of short war drives; a standard war-driving algorithm is applied to these fragments to estimate the locations of any beacons that were observed. These beacons are then added to the seed-set and are used to bootstrap the self-mapping for future days.

5.2 Self-Mapping with Public Beacon Data

Another source of seed data is the internet, namely public databases that share the location of 802.11 APs. This is probably the most compelling scenario, as no mapping effort is required by the user and his device does not require GPS. In this scenario, the user downloads public beacon information to his device. Each day the device is used, self-mapping is applied to the traces it has collected, using the public beacon information as a seed-set. We identified and evaluated two public sources of beacon location information: *war-driving databases* and *hotspot finders*, both of which are discussed below.

War Driving Databases

Sharing the discovered APs with the rest of the community is part of the war-driving culture. Many war-driving repositories exist, but the largest to date is http://wigle.net/ with over 2 million known AP locations as of February 2005. The data from wigle.net is publicly available and contains both the BSSID and location for the APs.

Of the 9450 unique 802.11 APs observed in our test traces, wigle.net contained 4000, or 43%. This is a significant portion of the observed APs which suggests that this data should work well with self-mapping. One issue, however, is that the wigle.net APs are primarily located near main roads and downtown/commercial areas, with little coverage in residential neighborhoods. As we will see in Section 6, this distribution has an impact on the accuracy of the self-mapping.

Hotspot Finders

A service commonly advertised to mobile workers is the *hotspot finder*: websites that list where both free and for-pay wireless networking can be obtained in a neighborhood. Hotspot finders are offered by web portals like Yahoo! (mobile.yahoo.com/wifi) and specialty companies like Jiwire (jiwire.com), and are all potential sources of seed data.

These hotspot finders list hotspot locations using street addresses rather than geo-coordinates. This issue can be managed by using an address translation service like MapPoint.net. A bigger issue is that these services list networks by the human-readable SSID (*e.g.* 'Northwest Coffee'), rather then the unique BSSID (*e.g.* 0F:34:9A:29:00:67). SSIDs are not guaranteed to be unique and it is common for franchised hotspots to assign all of their APs the same SSID. This creates a problem, as our algorithm must know the *unique* key that identifies an AP. One possible technical solution is to disambiguate APs with the same SSID using the structure in the self-mapping graph. This problem may also solve itself if providers begin exporting BSSIDs. For the purpose of these experiments, we assume that this problem will be solved, and we constructed a public hotspot data set using the following heuristic: from the complete set of APs observed in the test traces, we extracted those with SSIDs clearly indicating public access -- known large service providers like TMobile and Wayport, known local coffee shops and stores with public hotspots, and networks with suggestive SSIDs like 'guest' and 'public'. This yielded a seed-set of 750 APs, less than 8% of the 9450 observed in the traces. Even more so than the wigle.net data, these APs were heavily concentrated in commercial areas. This can be seen in Figure 2, which shows the position of seed set beacons (squares) and the self-mapped beacons (circles) for one of our test neighborhoods.

5.3 Self-Mapping with an Existing Radio Map

One final source of possible seed data for self-mapping is an existing radio map. Corporate IT departments generally have maps of their AP locations, as do many universities. The key characteristic of an existing radio map is its age. Since APs exhibit a fair rate of turnover, a radio map's quality will decay over time and can

Fig. 2. An overhead satellite map showing the results of self-mapping using the *hotspot finder* seed set. The squares mark the locations of the beacons in the seed set, while the circles denote the beacons placed by the self-mapping algorithm

substantially degrade in six months to a year. In this final scenario, we consider how well our self-mapping algorithm works with existing radio maps of different ages.

To simulate an existing radio map, we begin with the "control" radio map described in the next subsection, then artificially age this map by randomly dropping 50%, 75% and 90% of the known beacons to form different seed sets. This essentially models the uniform addition of new beacons over time. It does not model the movement or decommissioning of beacons. Since this random thinning is artificially uniform, we believe this measures the best-case performance of our algorithm.

5.4 A Control: War-Driving Radio Map

To separate the performance of beacon-based location from the performance of self-mapping for beacon-based location, we introduce a control scenario that uses war-driving results as the radio map used by the system. To construct a war driving radio map, we extracted the GPS-locked segments the traces from the all three volunteers (this amounted to over 300 hours of data). These trace segments were then fed through a war-driving program that estimated the position of each beacon by averaging the locations in which it was observed. Due to incomplete GPS coverage, this data set contained the locations for 6200 of the 9450 beacons that were observed during the eight weeks. Ideally, we would be able to do an explicit war drive of the test area to build a more complete control data set. This was not feasible however, as our volunteers covered substantial parts of the city as well as each taking at least one trip out of state during the test period. Despite its limitations, our war-driving map provides excellent coverage, containing a known beacon location for over 99% of the scans that contain at least one beacon.

6 Results

All self-mapping scenarios were tested in the following way: each volunteer's traces were divided by calendar days *1..n* based on the days the traces were recorded. For each day *i*, the trace was run through a location tracker that used that volunteer's radio map for day *i*. A volunteer's radio map for day *i* was constructed by running the self-mapping algorithm using the appropriate seed-set and the volunteer's traces from days 1*..(i-1)*. This models the situation that each night, the volunteer's device performs self-mapping with what it already knew, plus the new trace from that day.

- *Coverage* was measured by counting what percentage of non-empty radio scans contained at least one beacon that was in the radio map for day *i*.
- *Accuracy* was measured by comparing the GPS-locked portions of the trace from day *i* to the beacon-based location estimates produces by Place Lab's centroid tracker for the same time period.
- *Error* for a given radio scan is the distance between the location estimate given by the beacon-based location system and the location in the corresponding GPS readings. Error for a given day is the median error for all of that day's radio scans. The error for the entire test period is the mean of the daily median errors.

This approach to testing has a few shortcomings. First, it uses GPS as ground truth. Standard (non-differential) GPS has a median error of 8m, bounding the accuracy of our error estimates. Second, this approach can only estimate accuracy when the volunteer had a GPS lock. As most people spend a large portion of their days at either home or work, we made a special allowance for those places. Since the set of beacons seen in the same place is very consistent over time, we measured the geo-coordinates and set of beacons for each volunteer's home and workplace. This way, we knew the volunteer's location when he was at work or home, even if he did not have a GPS lock, thereby increasing the percentage of the time we could perform accuracy comparisons.

One final distinction remains to be introduced. Like most people, our volunteers spent a significant portion of their time in a few key places, most notably home and work. This made the performance of a particular scenario for a given volunteer heavily dependent on how it performs in those few places. In one respect, this is absolutely correct, since accuracy over a user's day is a very appropriate interpretation of 'accuracy'. Such a definition of accuracy could be said to be temporal, as it computes average accuracy over the time period of a day. Another reasonable approach, however, is to measure the accuracy over all the places a user goes, independent of how long he spends at each place.

To addresses these two notions of accuracy, we present two sets of numbers for each scenario: performance for the full traces that contain all of the data (including long periods of inactivity such as when the volunteers slept at night) and a set of *transit* traces. The transit traces are the full traces with sections removed in which the set of beacons seen by a volunteer was stable for more than two minutes. (Manual validation showed that this simple heuristic was highly effective at identified times when a volunteer stayed in one place). These transit traces represent the times that a volunteer spent walking around, commuting, shopping, etc.; arguably some of the most important times for a location system to perform well.

6.1 Accuracy and Coverage

Table 1 shows the accuracy and coverage for each volunteer in each scenario for full and transit logs. Accuracy is the average of the daily median accuracies; coverage is the percentage of the time the self-mapping radio map contained a location estimate for at least one beacon. We normalize the coverage numbers to the 90.2% average coverage that a complete beacon map could provide to show the coverage of the self-mapping radio map, not 802.11-based location estimation.

All three reality-inspired scenarios average between 84% and 96% coverage. This shows that there is sufficient connectivity in the self-mapping graph to estimate the position of a large fraction of the observed beacons. This is even true in the *hotspot finder* scenario in which the seed-set started off with less than 8% of the observed beacons.

Of the three scenarios, the best accuracy is provided by the *sporadic GPS* scenario. This is not surprising, as the occasional GPS readings cause the seed-set to grow over time, providing more information in the long run than the other scenarios. For V1 and V2, the accuracy of *sporadic GPS* is only 15% worse than the *war-driving* control

scenario on average. This is not the case for V3 who saw better accuracy with the *wigle.net* and *hotspot finder* seed-set than with *sporadic GPS*. This is due to the difference in GPS coverage between volunteers: V1 and V2 reported walking to work, and this is reflected in GPS coverage of 43% and 54% respectively in their transit traces. In contrast, V3 had only 24% GPS coverage in his transit logs. This does not mean that *sporadic GPS* is not a viable bootstrapping scenario for people like V3; it just means that it will take longer for their radio maps to reach war-driven accuracy.

Of the two public database scenarios, *wigle.net* performed better than *hotspot*

Table 1. This table summarizes the accuracy and coverage of self mapping for the test data. Each scenario from Section 5 is represented. This table also includes the accuracy and coverage of a war-driving map

	Transit Logs		**Full Logs**	
Scenario/User	**Accuracy**	**Coverage**	**Accuracy**	**Coverage**
Explicit mapping via war driving				
V1	21 m	-	9 m	-
V2	34 m	-	82 m	-
V3	33 m	-	79 m	-
Average	*30 m*	-	*57 m*	-
Self mapping with sporadic GPS				
V1	25 m	88%	10 m	88%
V2	47 m	84%	95 m	85%
V3	94 m	80%	124 m	79%
Average	*56 m*	*84%*	*76 m*	*84%*
Self mapping with wigle.net data				
V1	80 m	93%	829 m	89%
V2	72 m	97%	88 m	98%
V3	88 m	99%	110 m	98%
Average	*80 m*	*96%*	*342 m*	*95%*
Self mapping with hotspot finder data				
V1	174 m	94%	1450 m	88%
V2	106 m	90%	144 m	81%
V3	69 m	97%	54 m	96%
Average	*117 m*	*94%*	*550 m*	*88%*
Self mapping with random 10% of beacons				
V1	83 m	94%	600 m	90%
V2	124 m	94%	129 m	89%
V3	58 m	92%	57 m	87%
Average	*88 m*	*94%*	*262 m*	*89%*
Self mapping with random 25% of beacons				
V1	45 m	96%	32 m	92%
V2	49 m	98%	53 m	96%
V3	51 m	99%	117 m	98%
Average	*48 m*	*97%*	*67 m*	*95%*
Self mapping with random 50% of beacons				
V1	30 m	96%	25 m	93%
V2	33 m	88%	28 m	97%
V3	39 m	99%	93 m	99%
Average	*34 m*	*97%*	*49 m*	*96%*

finder in nearly every scenario. This was expected, as the *wigle.net* data set was bigger and largely a superset of the *hotspot finder* data. (Of the 750 APs in the *hotspot finder* data set, 510, or 68%, were also in the *wigle.net* data set.) These scenarios showed good coverage and accuracy of around 100m in the transit case. Recall that self-mapping only requires an 802.11 radio and involves no pre-mapping. Given the effort required of the user, these scenarios offer considerable potential.

Probably the starkest contrast is the difference in consistency of performance between the transit and full traces. In the transit cases, accuracy was typically within 50% across users, with a worst case of a factor of 2.5. In the full traces, nearly every scenario, including the war-driving control map, had more than a factor of 10 variation between some volunteers. This illustrates the critical effect that the placement of the dozen or so beacons situated around the volunteer's home and work have on overall accuracy. Part of this can be explained by where the volunteers lived. V3 lives in a downtown hi-rise building, and is in range of many more APs than V1 and V2 when at home. Accordingly, V3 had the best performance for full logs with the small *hotspot finder* seed set. Conversely, V1 lived in a residential neighborhood and was more than 500m from any of the APs in the *wigle.net* and *hotspot finder* seed sets, making his performance poor for the full traces. But the differences seen in the *war-driving* control scenario can only be explained by chance. All three volunteers had multiple beacons visible at home, and the war-driving placement was coincidentally very accurate for V1 and less so for V2 and V3.

This last observation highlights the effectiveness of the selective refinement technique used by Active Campus [4]. Active Campus allows users to override the radio model and click on a map to, in effect, say "Right now, I am here". This correction is stored in the radio map and supersedes the radio model in the future. Since the correction is based on a fingerprint, accuracy for these corrections should be 2-3m. Since our volunteers spent two thirds of their time at either work or home, correcting only these two places would have a drastic impact on average accuracy.

We now discuss the results for the last three scenarios in which a random 10%, 25% and 50% of the war driving radio map are used as a seed set for self-mapping. As expected, accuracy grows better as the size of the seed set is increased. Self mapping with the 10% seed set provides accuracy three times worse than war-driving in the transit case, while the 50% seed set is nearly as good. This shows that existing out-of-date radio maps can make excellent seed sets for self-mapping.

By dropping random beacons from the war-driving data set, we have created a uniform, random distribution. This provides an interesting opportunity to compare the non-uniform *wigle.net* and *hotspot finder* seed sets to these uniform seed sets. Despite the fact that the *wigle.net* seed set contained over 40% of the observed APs, its performance was closest to the 10% random seed set. Similarly, despite the fact that the *hotspot finder* seed set contained close to 10% of the observed beacons, its accuracy was considerably worse than the 10% random seed set. Intuitively, adding a new beacon in an area with no other known beacons should add more value than adding one in an area already populated with known beacons. This data illustrates that the even distribution of a seed set is as important as its size.

Finally, Figure 3 shows the effect that the number of neighbors that a node has in the graph has on the accuracy of beacon placement. These graphs show aggregate results from all three volunteers in all scenarios. The y axis on both graphs shows the

distance between the war-driven position of a beacon and its placement by the self-mapping algorithm. Assuming the war-driving estimate is accurate (we validate this assumption in Section 6.3), this is a measure of the self-mapping algorithm's accuracy at placing beacons. On the graph on the left, the x axis shows the number of neighbors, or edges that a beacon's node has in the self-mapping algorithm. As the number of neighbors grows, the location estimates grow more accurate. This fits the intuition that more constraints yield more accurate solutions. The graph on the right of Figure 3 shows how the accuracy of beacon placement varies with the number of *anchored* neighbors. This graph shows a pronounced improvement in accuracy when going from 0 to 1 anchored neighbor, meaning that inferring a beacon's location from observations containing known beacons yields much better results than doing a chained inference through other non-anchored beacons. Beyond the first three anchored neighbors, little to no extra accuracy is gained. This further reinforces our claim that seed sets with clusters of beacons within range of each other will not yield as accurate a radio map as a similar sized seed set with a more uniform distribution of beacons.

6.2 Performance Over Time

To show how self-mapping works over time we present a set of time-series graphs in Figure 4. These graphs show results for three scenarios (*sporadic GPS*, *wigle.net* and *random 25%*) for the transit traces for V2. Every graph is a time series over the 31 days of V2's traces. Each scenario is shown as two stacked graphs. The top graph in each case shows the coverage offered by the self-mapping radio map that day and the cumulative percentage of beacons that were mapped to that point. The bottom graph for each scenario shows the median accuracy for that day and includes the median accuracy for the war-driving control data set to serve as a comparison.

Since the *sporadic GPS* scenario starts out with an empty seed set, its coverage and percentage of beacons mapped start at zero and ramp up more steeply than the other scenarios. The most drastic feature of the coverage graphs is the large drop that occurs during days 6-8. This dip corresponds to a trip that V2 took out of state. The dip is most pronounced in the *sporadic GPS* scenario as V2 had no mapped beacons on

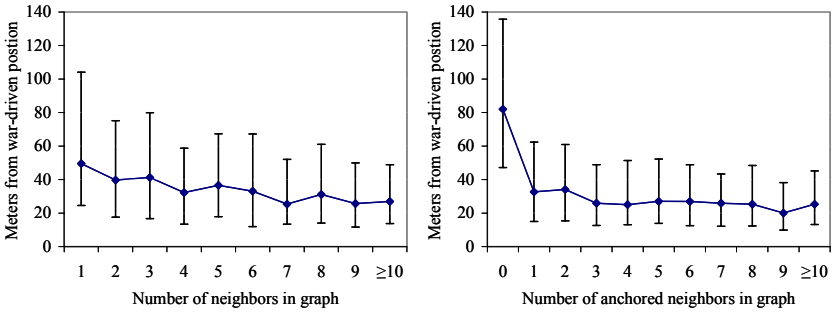

Fig. 3. These plots show how the structure of the self-mapping graph affects the accuracy of beacon placement. The left plot shows how accuracy varies with the number of neighbors. The right plot shows how the accuracy varies with the number of anchored neighbors

arrival. During these three days, the coverage increases each day as the self-mapping algorithm places beacons in this new locale. Upon returning home, the coverage resumed more or less where it had been before V2 left town. A similar, but less pronounced dip can be seen on days 23-25 when V2 spent portions of his day visiting

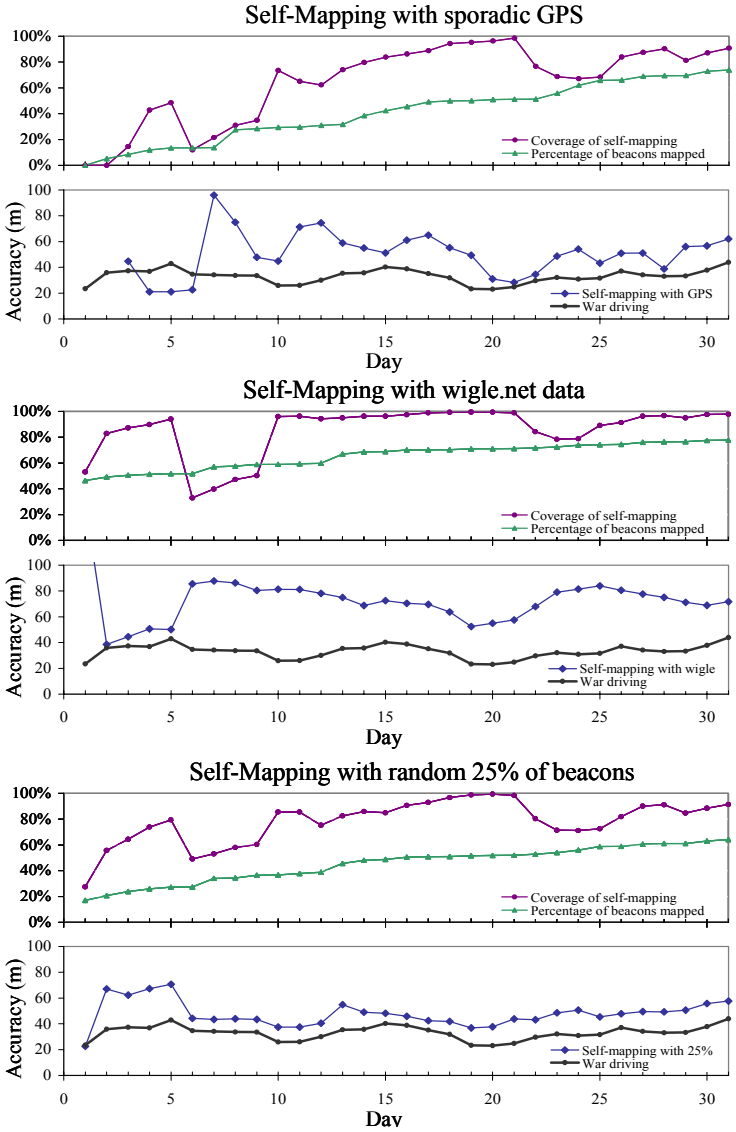

Fig. 4. Timelines comparing the coverage and accuracy of three self-mapping scenarios for V2's transit data. For each scenario, the upper graph shows coverage provided that day and percentage of total beacons mapped. The lower graph compares the median accuracy provided by the self-mapped radio map with the war-driven map

Table 2. A comparison of the accuracy of self-mapping and war driving beacon placement

Error in AP Placement (Meters)		
Access Point	Self-Mapping	War Driving
00:09:d7:c4:3c:81	35	27
00:09:5b:99:a9:c0	54	37
00:04:5a:0e:6e:fc	24	32
00:02:3a:9e:a3:d7	11	14
00:0f:3d:4f:84:a0	31	18
Average	**31**	**26**

places that were not yet mapped. Note that for each of these periods of low coverage, there is a corresponding bump in the percentage of beacons mapped; visiting new places results in periods of poor coverage, accompanied by an increase in the size of the radio map.

We were pleased by how quickly the accuracy stabilized. Our self mapping algorithm reaches its peak accuracy after only a visit or two, causing accuracy to stabilize in days rather than weeks. This suggests that in real use, users would start to see improvements in accuracy and coverage quickly.

6.3 Keeping a Radio Map Fresh

Since a key benefit of self-mapping is the incorporation of new beacons, we wanted to quantify the accuracy of the location estimates of new beacons. To simulate newly deployed beacons, we chose five beacons from the war-driving radio map described in Section 5.4 and manually surveyed their locations. (One beacon was in the home of an acquaintance, while the others were in publicly accessible businesses.) The comparison of the manual survey and the self-mapping are shown in Table 2. For each beacon, we removed it from the radio map and ran the self-mapping algorithm using V1's traces. We then measured the error in the location predicted by the self-mapping algorithm. For comparison, we also calculated the error in the original war-driving placement that was used as ground truth in Section 6.1.

Table 2 shows that given an up-to-date radio map, the self-mapping placements are almost as accurate as war-driving. For these APs, the self-mapping placement was less than twice the error of the war-driving placement and was within 5m on average. Thus with reasonable beacon density and an up-to-date radio map, self-mapping can add new beacons with nearly the same accuracy as war-driving.

7 Conclusions

We have presented a graph-based, algorithm that allows a small set of known beacon locations, or *seed data*, to be expanded as users naturally use the system. We believe self-mapping is a key to enabling radio-beacon based location become a widely used high-coverage, indoor/outdoor location technology. We have presented three realistic scenarios for seeding our algorithm, each of which we tested with nearly 100 days worth of traces containing over 20 million readings. Our results have shown that:

1. Due to the structure present in radio scans, even small seed sets can be used to bootstrap a self-mapped set of radio locations that provides high coverage.
2. The accuracy of self-mapping is highly dependent on the distribution of the beacons in the seed set. Our results showed that a uniformly distributed seed set performed as well as a clustered seed set four times as large.
3. Both public access point databases and sporadic GPS coverage are viable, real-world sources of seed data. Neither requires users to explicitly perform mapping, and both showed the ability to ramp up to reasonable coverage within a few days of normal use.
4. With as little as 50% of the beacon locations, self-mapping can produce a radio map that estimates a user's location as well as a war driving database. Further, we showed that when new beacons are introduced, self-mapping estimates their positions nearly as accurately as war driving.

8 References

1. BAHL, P. & PADMANABHAN, V. RADAR: An In-Building RF-Based User Location and Tracking System *Proceedings of IEEE INFOCOM*, pp. 775-784 (2000)
2. DOHERTY, L., PISTER, K. S. J. & GHAOUI, L. E. Convex Optimization Methods for Sensor Node Position Estimation, *INFOCOM 2001*.(2001)
3. GIROD, L., BYCHKOVSKIY, V., ELSON, J. & ESTRIN, D. Locating Tiny Sensors in Time and Space: A Case Study, *ICCD 2002*.(2002)
4. GRISWOLD, W. G., SHANAHAN, P., BROWN, S. W., BOYER, R., RATTO, M., SHAPIRO, R. B. & TRUONG, T. M. ActiveCampus - Experiments in Community-Oriented Ubiquitous Computing, *To Appear: IEEE Computer*.
5. HAEHNEL, D., BURGARD, W., FOX, D. & THRUN, S. An Efficient FastSLAM Algorithm for Generating Maps of Large-scale Cyclic Environments From Raw Laser Range Measurements *IROS 2003* (IEEE/RSJ).(2003)
6. HE, T., HUANG, C., BLUM, B. M., STANKOVIC, J. A. & ABDELZAHER, T. F. Range-free localization schemes for large scale sensor networks, *MOBICOM 2003*.(2003)
7. KRUMM, J., CERMAK, G. & HORVITZ, E. RightSPOT: A Novel Sense of Location for a Smart Personal Object, *UbiComp 2003*.(2003)
8. KRUMM, J. & HINCKLEY, K. The NearMe Wireless Proximity Server, *UbiComp 2004*.(2004)
9. LAASONEN, K., RAENTO, M. & TOIVONEN, H. Adaptive On-Device Location Recognition, *Pervasive Computing 2004*.(2004)
10. LAMARCA, A., CHAWATHE, Y., CONSOLVO, S., HIGHTOWER, J., SMITH, I., SCOTT, J., SOHN, T., HOWARD, J., HUGHES, J., POTTER, F., TABERT, J., POWLEDGE, P., BORRIELLO, G. & SCHILIT, B. Place Lab: Device Positioning Using Radio Beacons in the Wild, *Pervasive 2005*, Munich, 2005.(2005)
11. PRIYANTHA, N. B., BALAKRISHNAN, H., DEMAINE, E. & TELLER, S. Mobile-Assisted Localization in Wireless Sensor Networks, *IEEE INFOCOM*, March.(2005)
12. SEIDEL, S. Y. & RAPPORT, T. S. 914 Mhz Path Loss Prediction Model for Indoor Wireless Communications in Multifloored Buildings, *IEEE Transactions on Antennas and Propagation*, 40, 207-217. (1992)
13. WANT, R., SCHILIT, B., ADAMS, N., GOLD, R., PETERSEN, K., GOLDBERG, D., ELLIS, J. & WEISER, M. The ParcTab Ubiquitous Computing Experiment, *Mobile Computing*, pp. 45-101.(1997)