The Design of the Mirage Spatial Wiki

Nels Anderson, Adam Bender, Carl Hartung,
Gaurav Kulkarni, Anuradha Kumar, Isaac Sanders
Dirk Grunwald and Bruce Sanders
Dept. of Computer Science
University of Colorado

Keywords: Location based systems, ubiquitous computing, wiki, Internet systems.

Abstract: Location based services can simplify information access but despite the numerous efforts and prototypes that attempt to provide location based services, there are very few such systems in wide spread use. There are three common problems that face the designers of location based systems - the basic location technology, the complexity of establishing a service for a particular location and the complexity of maintaining and presenting information to users of the systems. This paper outlines the software architecture of and experience with the Mirage Spatial Wiki. We describe the design decisions that have led to a system that is easy to deploy and use.

1 INTRODUCTION

This paper describes the software architecture of and experience with the Mirage Spatial Wiki, a location based information system that is easy to deploy and use. Location based services are designed to simplify information access by using the current location to direct the user to relevant information. There are many examples of such systems (we discuss many of them in §5) and almost as many goals for those systems. Some location based systems are intended to be globally accessible using a variety of access devices. These systems commonly use a combination of GPS satellite positioning coupled with other systems for indoor positioning and assume data access is *via* existing wireless networks or cellular data networks such as GPRS. Other location based systems depend on a specific infrastructure, such as location systems for cellular phone networks, which rely on proprietary RF detection for location and provide service via WAP or other protocols designed for mobile phones. These systems are intended for large-scale commercial deployments and provide services such as way-finding, location-specific shopping information or tour-guides for cities (Cheverst et al., 2000). Some other location systems are intended for smaller scale deployments that are focused on more specific business needs, such as locating resources (*e.g.* printers, meeting rooms, *etc*) within a specific building or providing an automated docent as a tour guide(Hightower and Borriello, 2001). Again, these systems commonly use a variety of location detection systems, ranging from infra-red or ultrasound "beacons" to radio network signal strength. Often, these systems rely on specialized hardware components or large or complex software environments and are tightly integrated to specific applications.

Systems involving geo-spatial coordinates, such as those obtained from GPS satellites, are difficult to integrate with approaches that use beacons; these latter systems are commonly referred to as *semantic* or *proximate* location systems. Their goal is to provide information about objects in a particular reference frame, not to establish a mapping of an object in an absolute coordinate system. Depending on the intended application, knowing that the nearest fax machine is "*down the hall in room 422*" may be more meaningful than knowing the specific latitude/longitude of the device. Even in the presence of a system to assign objects locations in a coordinate space, the maintenance of such a mapping is complex and possibly pointless in environments where acquiring geo-spatial information is difficult or impossible (such as in buildings when using current GPS technology).

This paper describes the software architecture of a system we developed to provide location based services in educational environments. The system is based on past experience with "beacon" based location systems similar to the infrastructure used in the

Cooltown system. The difficulty in deploying the previous system guided the design choices and compromises made in the eventual system. Overall, the system relies much more on Internet enabled applications and seeks to maximize the existing infrastructure. The resulting system is very easy to deploy, very scalable and easy to maintain. We have demonstrated that we can "map" a large building in less than two hours and provide rapid access to a "spatial wiki" that simplifies the sharing of location-based information.

In this paper, we provide an overview of the system in §2 and then discuss specific details of the system in §3. We compare the system to prior work in Section 5 and summarize our findings in §6.

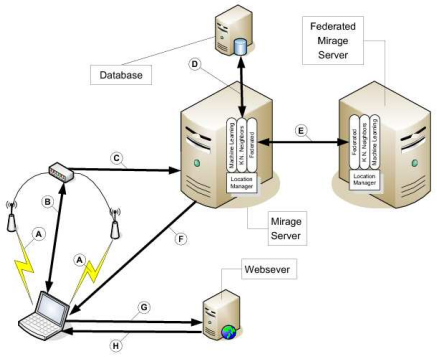

Figure 1: Components in Mirage Location Service and their interaction during a query. The laptop receives station beacons from the local access points (A), and uses the Mirage desktop locater to launch a web browser that relays the access point information through the network (B) to the subscribed Mirage server (C) as an HTTP request. The subscribed Mirage server uses uses the different location blades to resolve the request; the blades communicate with the database (D). If the location blades can not resolve the query, the subscribed Mirage server can forward the request to a federated Mirage server (E). If no location is determined, the default location is used. The URL corresponding to the resolved location is returned to the laptop as an HTTP redirection (F). The laptop then contacts the webserver specified in the HTTP redirection (G) and is given the actual content corresponding to the location (H).

2 SOFTWARE ARCHITECTURE

As part of a broader educational program, we wanted to deploy a location based system in a university setting to provide information about laboratory equipment, contact information for people in a specific department as well as class-room specific information such as schedules and an electronic "lost and found". One of the over-arching goals for our design was to provide a system that allowed the users of the system to easily develop new uses for the system and for the system to require little maintenance. In our initial experiment, we used infrared beacons to identify rooms and depended on an existing 802.11 wireless network to provide network connectivity. This system proved to be unwieldy because the number of beacons needed was prohibitive and the beacons were difficult to use in practice since infrared signals are fairly directional.

Based on our earlier experience, we developed a system that would address the following goals:

- It should not require additional system hardware components to be installed,
- It should support the targeted applications (lab specific information, *etc*)
- It should be extensible enough to enable new applications,
- It would be easy to maintain,
- And, it should work with common portable computer operating systems,

The resulting system, called the Mirage Spatial Wiki, uses "beacons" provided by a ubiquitous 802.11 network to provide a semantic location service. Standard 802.11 networks have limited range in most buildings and robust installations of such networks typically deploy numerous access points across multiple frequencies or channels to provide sufficient coverage and bandwidth. The beacons from these individual stations can be used to provide approximate location; using these beacons has the side benefit that it simplifies the network design – access points not only provide location information, they also provide the data network needed for indoor location based services.

Earlier systems used "RF triangulation" (Bahl and Padmananbhan, 2000) to approximate a geo-spatial location service; our own experience with this approach resulted in failure, because RF signals exhibit signal fade and significant multipath characteristics in many institutional buildings, particularly those made of concrete. When using RF triangulation, we found that the location would rapidly "jump" from one spatial estimate to another. Moreover the system required considerable calibration and training to provide enough information for the RF triangulation system, and calibration was needed when furniture was moved around or even when doors were left open rather than closed. Thus, rather than focusing on determining precise locations, we emphasized the notion of semantic locations defined as different "*land-*

ings" that had some external significance. For examples, a classroom, a portion of a hallway or lobby or a specific floor in a building. By shifting to semantic locations, we could use different location calculation systems that traded geo-spatial accuracy for the ability to correctly identity the coarser-grained semantic space. We eventually found two algorithms that worked well when using noisy data sources such as the RF signal strength from 802.11 network beacons. The first algorithm, based on the K nearest-neighbor matching algorithm, works well with limited training data and provides sufficient accuracy to give room-level locations. The second algorithm, based on a back-propagation neural network, provides more accurate in-room positioning, but requires considerably more data to achieve such accuracy.

Both of these semantic location algorithms require training data. Although some location based systems, such as PlaceLab (Schilit et al., 2003), distribute that data to client computers, that design appeared to be overly complex. There are three motivations for using client-side location data. The first is to provide location information even when a network connection to a centralized location server is not available. The second is to insure that the location system is scalable as the number of clients increases and the third reason is to enhance privacy in location based systems. Although these design goals are laudable, they're less important for the system goals we envisioned. First, since our location based system relied on a network to access location-based applications, there was little point in having location information without network connectivity. We felt that scalability could be addressed using conventional network load balancing techniques, and developed a design that allows us to partition the components used in the full location based application. Lastly, although privacy is an important issue, the privacy-revealing information exposed by network connections would not be directly visible to the location system and there are other ways to disguise such information (Gruteser and Grunwald, 2003).

The resulting system relies on a small desktop application that uses the underlying operating system to record information from available access points, including the hardware MAC addresses, the network names (SSID) and received signal strength. Due to the design of the 802.11 protocol, this information can be collected from all available networks, including those that use security protocols or to which the user does not actually have access. We developed similar applications for Windows/XP, Mac OS-X and two common Linux desktop environments (Gnome and KDE). In each case, the application records the information and launches a web browser with a URL request directed to a web application on a specific Mirage server.

The Mirage server determines the closest "landing"

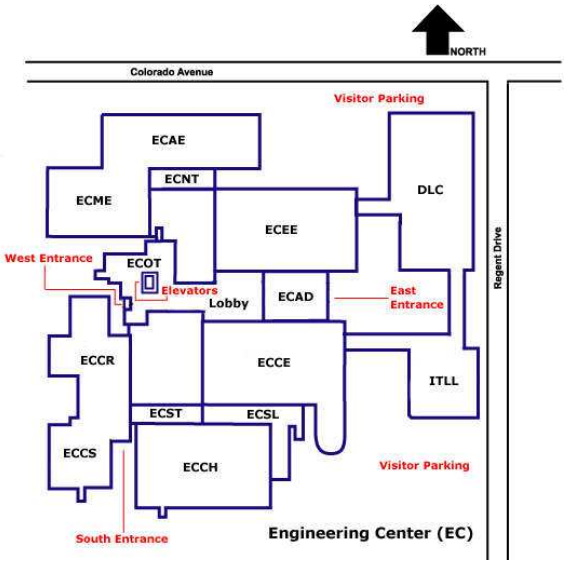

Figure 2: Example Deployment Scenario

corresponding to the location information using one or more of the available "location blades". Each "location blade" may report a degree of certainty in a particular decision. If the location can not be determined using local information, the specified Mirage server may communicate with other Mirage servers using an optional federated network architecture.

There is a corresponding URL associated with each "landing"; using the HTTP protocol, the Mirage server directs the client browser to the specified URL. Figure 1 illustrates the sequence of operations. As illustrated, the system is composed of three distinct parts (the location server(s), the database for location information and the web server for the location based applications), but all of these components can be hosted on a single system if desired.

Rather than develop a complex application framework, it is possible to use existing Web applications in conjunction with the Mirage location system. One of the continuing challenges facing any information space is the need to provide the information needed by the users of that space and to insure that information is up-to-date. Our initial system uses a WIKI (MediaWiki-Foundation, 2004) collaboration tool to simplify maintenance of these information spaces.

Each "landing" is represented by a page in the Wiki, and the content of that page can be edited by users of the system. A sample page is shown in Figure 3. In our application, each landing includes a schematic map for the region and includes links to adjacent landings. That map is added as a hand-edited graphic indicating the intended physical space represented by the "landing". Figure 2 shows the sample building used in the initial deployment. The build-

ing is the primary building in an engineering department. The building is divided into different departments with a range of physical space represented by each department; each labeled department is a "landing" in our current deployment. Not shown in the building is an office tower that occupies the region labeled ECOT; each floor of that 8 story structure is also a "landing". Currently, each "landing" occupies considerable physical space, but this is an artifact of our prototype system. We have demonstrated that we can subdivide a small physical region into a number of "landings" that measure $\approx 100 \times 100$ sq. ft. For example, the region labeled "ECCS" can easily be subdivided into four distinct regions.

The trade-off between the granularity of a "landing" and the certainty in identifying which "landing" the user may be visiting is a key compromise in the system we designed. Unlike earlier systems that attempted to provide meter-level resolution of physical location, such as Radar, we opted to address larger regions and provide navigational and training mechanisms to allow the user to locate their true location from the list of possible locations. The long-term effectiveness of this technique can not be assessed without a longitudinal usability study, but our *ad hoc* assessment is that it provides considerable utility – rather than force the software to be completely accurate, we push part of the responsibility back to the user. Overall, the system provides a usable location based service for the intended applications.

3 DESIGN DECISIONS

The Mirage system relies on the ability to represent physical location using information gleaned from wireless networks. Using three of the more common pieces of information readily available from operating system calls: MAC address, Signal Strength, and Signal Set Identifier (SSID) we are able to form a framework in which we can represent a unique location as a combination of these attributes. To better understand how the Mirage system is capable of deducing location it is beneficial to follow the life of a query in the system.

Upon launching the desktop client the underlying system calls are made to query surrounding wireless networks, the three quantities necessary to represent location are gathered and parsed into an appropriate URL on the Mirage system. Once the query reaches the Mirage system a series of checks are made to verify that the MAC addresses and their corresponding SSID's have been seen together before. This is a crude check to warn Mirage of possible change to the physical layout of the network (i.e. an access point was moved or renamed). Since the SSID is the most

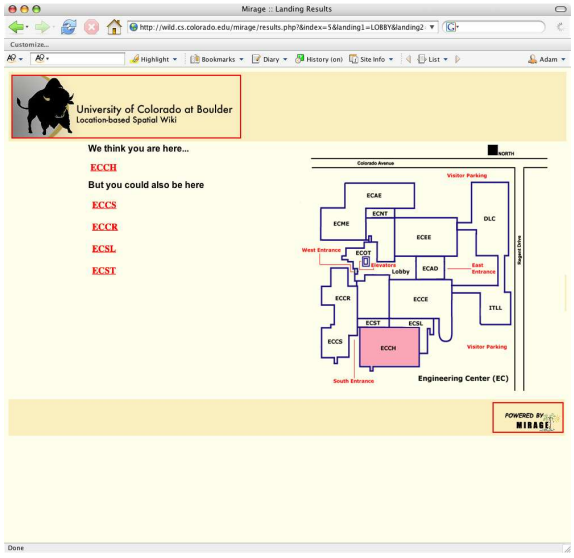

Figure 3: Screen Capture Showing Location

arbitrary quantity in the set this is by no means a guarantee but rather a hint that Mirage may not perform as accurately as before. Once the verifications have been made an "encoding-pair" scheme is formed in which each MAC address is attached to its corresponding signal strength and the total number of encoding pairs being submitted is pre-pended and the query is passed to the first "location blade", the K Nearest Neighbor module.

Mirage currently makes use of two separate and well-known machine-learning algorithms to help make a determination of location. The first of those and the more general of the two is a weighted K Nearest Neighbor (KNN). The reason for choosing KNN is two fold: **a)** There is no training time to build a model, and **b)** It can easily produce a ranking system that provides feedback on what else is nearby.

The KNN algorithm depends on the definition of a metric in a specific "feature space"; in general this can be a significant challenge, however for our system it is a straightforward hashing. Using each MAC address as a unique offset into a vector, Mirage encodes the signal strengths into an n-dimensional vector, where n is the number of unique MAC addresses that have been seen by the desktop client. It is well documented that the KNN model is decidedly ineffective if the feature space has more than ten dimensions (Russell and Norvig, 2002); in the case of Mirage we may have a space of 1000 or more dimensions corresponding to 1000 unique MAC addresses in the deployment area. Thus far in trials, we've observed ≈ 100 unique MAC addresses on single floor of a building. However, these vectors are very sparse, having only 4 or 5 non-zero entries; again in practice

the most we have ever observed on a single query is 7 non-zero entries – *i.e.* seven unique access points. It is a unique feature of this kind of data that has an almost implicit feature reduction mechanism; it is unlikely that network administrators would install many more wireless access points to compete for the three available wireless channels.

Once the query vector has been encoded it must be compared with the training set data. The training data is pulled from the database server and formed into vectors using the same hash that created the query vector. It is important to note that in order to reduce the size of the "online training set" (the set of data actually used to compute a given location) and therefore the overall computation time for a given query, it is necessary to prune the training data to include only those entries which are "nearby". For a training item to be nearby it must have at least one MAC address in common with the query, this helps prevent some data items which may have generally low signal values from throwing the system off and also validates the use of the metric. Once the online training set is complete the metric is applied to each item in the set and a distance is computed between it and the query. In this case the metric used is the standard Euclidian distance, with a slight modification for weighting. In order to give more weight to a MAC address that the query and the training item have in common we use a multiplier to reduce the value of the term. In effect this helps to correct for outlier signal values, which may be very weak and so undercut the distance calculation. The results of all the distance calculations are then sorted in ascending order and the first K values are selected from the list. In general K is an odd number and is usually less than ten. Mirage uses a K of three that was derived *via* empirical testing. Each training example is also tagged with a classification; in this case it happens to be a "landing". Using the first K values, a majority computation is performed to determine the most common classification of the K nearest neighbors. This is the value assigned to the query. In the case of a three-way tie the first or closest neighbor is used as the classification. This process then begins again, removing all training examples from the online data set with the classification initially assigned to the query and proceeds until all the classifications have been assigned a ranking. Upon completion a set of classifications is produced in order of proximity to the query. These results are then returned to the web server and the appropriate results web page is generated, as shown in Figure 3.

We specifically chose to limit the number of returned locations because our initial testing indicated that users were less receptive to a system that presented a large number of alternative locations – they felt the system was defective rather than realizing the importance of the probability of a particular location estimate.

The generation of the base training set for use with KNN is a very quick process. Using an automated desktop client that uses the same database interface as the end user version, an administrator can collect wireless network data at a specified interval (i.e. every two seconds during a scan) and submit the information via a web based mechanism. The administrator simply indicates the current "landing" and then walks through the physical area corresponding to the landing. Once the data reaches the Mirage server it is parsed into encoding pairs, tagged with a "landing" classification and inserted into the database. The data is then immediately ready for use by the next call of the Mirage system.

The second level of location detection in the Mirage system is done using a back propagation neural network. This technique can provide a significantly finer granularity in terms of distance than KNN (in practice the neural net model has achieved accuracy on the order of $\pm 1.5m$), however there is a significant overhead associated with training the network. We modeled the neural network algorithm on a similar project at the University of Trento, Italy (Battiti et al., 2002) and choose a three layer, fully connected neural network structure. A fully connected neural network implies that every node in layer n is connected to every node in layer $n + 1$ except for the output layer, which has no upward connections. The input layer size is equal to the number of unique MAC addresses in the nearby vicinity. This is potentially a much smaller set than that which is used by the KNN module, because the system has already performed a level of determination. It is only necessary to include MAC addresses present in training examples with the same classification. This also means the network will have to learn a much smaller variance with regards to the input layer. The hidden layer of the network consists of ten nodes; this parameter was derived empirically and further investigation may prove to change its size. The output layer of the network consists of two nodes, one representing the x coordinate and the other representing the y coordinate. The result which is presented at these outputs will represent the models prediction of the physical location in (x, y) space of the user that generated the query. In order to build the model, the network must be trained on a set of data specific to the physical location it is to model. Each "landing" as specified by the KNN module will have a corresponding neural net that has been trained specifically on examples that are considered to be physically connected to it. Of course what belongs to a certain landing is somewhat at the discretion of the administrator who first trains the system. The data itself is very similar to the data used in the KNN module (i.e. encoding pairs) except that the landing classification is now an (x, y) coordinate pair.

This is perhaps the biggest draw back to the neural network approach, as it requires a very extensive training set. Once a training set has been loaded, the network is presented each example and the signal is propagated to the output layer, the deviation of the output and the stored classification is determined using Euclidian distance in the plane. This error signal is sent back down through the layers and used to adjust the weights in the network. This process is repeated for each example in the training set and a running total of the squared error is kept to monitor the network progress. The training is terminated once the error rate per pass through the data or "epoch" has crossed below a minimum threshold. At this point the network structure is stored and can be used for classification of new examples. The neural networks generally train within a minute or two but can have a training time as high as ten minutes depending on training set size. Once the model is built and stored, queries can be passed to it and encoded as a vector. This vector is then presented to the input layer of the network; the resulting signals are propagated upward to the output layer and produce a set of coordinates. These coordinates are then passed to the web server and used to generate a location on the floor plan.

Although the neural network system can provide fairly accurate location determination, it is still sensitive to vagaries of RF propagation. In general, we've found that the users of the prototype prefer to be presented with information that provides a higher assurance of their location and which facilitates navigation to the specific items of interest. Thus, rather than attempting to determine *e.g.* the specific location of a user in a hallway, it is better to display the whole hallway and allow the user to zero in on the specific room of interest. Since the rapid KNN classification system is able to accurately determine location for $\approx 100 \times 100$ ft. regions, this places little practical burden on the user.

4 EXPERIMENTS

To examine the accuracy of Mirage, we designed several experiments to populate the system with varying numbers of training data points for different locations. We then launched the system at different positions within the trained location and recorded the rank returned by the system. A rank of 1 means the system returned our correct location, whereas a rank of 2 means the system thought our location was 2nd most likely. All training data for these experiments was entered and taken using an Apple iBook. All training data points were taken at 2 second intervals walking down the centers of hallways, or across rooms.

Our first experiment was in a seven story office tower to examine the effects of physical barriers between landings. For floors 2-6 and 8 we used 90 points of training data per floor in order to create existing landings for our experiments. On floor 7 we trained the system using 30, 45, 60, and 90 data points. For each level of training, we then launched the system at 10 different positions and recorded the rank given to the 7th floor. For each of the 30, 35, 60, and 90 data points our system returned an average rank of 2.0, 1.7, 1.8, and 1.2 respectively. We conducted an ANOVA analysis comparing the importance of the number of training samples *vs.* the wireless card used to evaluate a location. That analysis indicated that the amount of training was more significant than the type of card used, but the predictive accuracy of the ANOVA model was very low. Given The "mean rank" across different wireless cards ranged from $1.2 - 1.7$.

Next, we used the building's lobby to test landings without physical barriers. We used the system to divide the lobby into 3 logical zones (1 larger, and 2 smaller) which did not have any physical barriers. We trained the two smaller areas with 60 data points and the large area with 90. We then proceeded as before and took 5 readings in each area, moving progressively toward the invisible borders of the areas. Our system returned an average rank of 1.0 for all three partitions, indicating that our system is accurate even in the absence of physical barriers.

Our final experiment addressed many small landings in close proximity to one another. We trained four smaller classrooms on the edges of a large classroom. The smaller classrooms were trained with 20 data points while the larger classroom was trained with 30. We then took readings in two of the smaller classrooms and the one larger classroom in random locations. Once again, our system returned an average rank of 1.0 for all three of the classrooms, showing that our system is accurate in the presence of many landings in close proximity.

From our experiments we can see that physical barriers tend to affect the accuracy of the system. This makes sense since radio propagation can be greatly affected by shadowing affects produced by different objects. However, even with such variances our system was highly accurate; we think the accuracy could be improved by training using a number of different brands of wireless cards and using that card-specific data for location queries.

5 PRIOR AND RELATED WORK

The benefits of location detecting systems have been well explored and documented over the past decade or so. The advent of systems such as GPS and

the cellular telephone network have allowed a multitude of applications to be developed using their wide infrastructure. However, there are several downfalls with the previously mentioned technologies. GPS, for example, has trouble providing accurate readings indoors, and both GPS and cellular networks require specialized hardware and software to be able to use. These problems have led to the development of a handful of systems which attempt to use more common hardware and ubiquitous infrastructures to provide location information.

Many systems now use 'beacons' such as 802.11 wireless access points and cellular towers, to help determine location. In all of these systems, the beacons must be recorded (Byers and Kormann, 2003) in some fashion in order to determine location. An example of such a system is PlaceLab (Schilit et al., 2003). The PlaceLab client notes the current beacons which it sees, and then checks a previously populated database to attempt to determine the user's location. The database is pre-populated either with location information provided by the owner of the network, or by "*war-driving*" where a user travels through a city with a GPS unit marking the location of any seen beacon. In either case, previous access point location information must be known and entered into the system before the client can report accurate data. However, the benefit of such a system is that it works indoors as well as outdoors.

PlaceLab was also designed to provide location detection in the absence of a data network. In practice, installing PlaceLab on a laptop requires installing an SQL database, a webserver and desktop clients. By providing "client side" location information, PlaceLab not only provides location services in the absence of network connectivity, it also preserves anonymity by not exposing location information or queries to centralized systems.

A few other specifically indoor location systems include RADAR (Bahl and Padmananbhan, 2000) and Cricket (Priyantha et al., 2000). RADAR uses special base stations that process the signal strength information of RF signals. The locations of the base stations must be previously known for this system, and the system requires extensive training. In our own implementation of a system similar to this, we found that the training data was very sensitive to room and building configurations. Each major change in that configuration (*e.g.* moving a metal filing cabinet) required retraining. In part, this was caused by the over-specification of the final location; in many cases, users are willing to trade precision for accuracy. Similarly, the Cricket location system uses special beacons that emit ultrasonic pulses combined with radio signals to determine distance measurements. Again, the location of the beacons must be previously determined and a mapping must be established between geo-spatial coordinates and the semantic coordinates of the tagged items.

The Cooltown system (Kindberg et al., 2002)(Caswell and Debate, 2000) uses infrared beacons to mark specific locations and regions. An earlier implementation of our system used Cooltown "beacons" to designate locations at the level of individual rooms. We found that maintaining the beacons was cumbersome. Moreover, considerable software was needed to detect and react to the beacons. One benefit of IR-based systems such as Cooltown is that the system can determine the direction the user is facing and then use that information to guide the user. However, the complexity of guiding users to point their IR receivers in the intended direction is considerable; we theorize that it would be better to simply present a photo to the user indicating the direction that they *should* face to make sense of the information presented. We plan on evaluating this decision for a museum docent we are deploying.

Another system utilizing only 802.11 access points is the ActiveCampus system (Griswold et al., 2002). ActiveCampus uses the reported RF signal strength from currently seen access points and infers the user's location by a least-squares fit algorithm; the location of the access points must be known in advance. However, the ActiveCampus system provides the location service independent of an underlying application framework. This means the locations can be used by applications such as "chat" or "graffiti", but it also means that there is little support for casual or *ad hoc* use of location information.

Our system differentiates itself from the above approaches by not necessarily needing to know the exact locations of access points, or know their locations in advance. If a user comes across an unknown access point, they can create a landing at that location which other users can connect to later. Also, our system learns from use where specific access points are, and never needs to be given strict location information about any access point. Furthermore, our system requires no special hardware because it uses the existing 802.11 infrastructure. Conveniently, this infrastructure is being deployed rapidly in a large percentage of places where people spend most of their time.

In addition, our system protects a user's privacy while still being extensible in such a way that if a user wishes to reveal more information about him/herself, other features are possible such as a buddy list or 'last seen at' person finder.

6 CONCLUSION

Overall, our experience with Mirage has demonstrate the tradeoff between accuracy and usability as

well the necessity of using existing Internet applications where possible. At the current time, it is possible to "map" a sizable building in a few hours and to provide meaningful location-based content in a similar amount of time. This implies that location-based systems may be able to move from being academic "science projects" to being usable systems that address the needs of an increasingly mobile workforce. It also opens new applications for location based systems, including meetings, conferences and other temporary events.

Both the K Nearest Neighbor and neural network models appear to offer advantages in the Mirage system. The KNN system has a very low setup time and can become accurate with a very sparse training set. It also can be updated rapidly and facilitates rapid deployment of location base information space. Unfortunately, it is not able to pick out the subtle differences once the distances become very close together, such as in the same room. On the other hand the neural network approach requires a high overhead of training examples and training time so it is not nearly as easily configured. However, once in place it can produce extremely accurate results. Current investigation is being directed at producing very high-resolution training sets for use in the neural network to possibly extend accuracy even further and the corresponding work on determining how to incorporate that more accurate information into meaningful location based applications. In either case the Mirage system provides an ensemble approach to location determination that is capable of satisfying both the "quick and dirty" set up and the much more finely tuned requirements.

The modular structure of the "location blades" should allow us to interface to different data sources to provide useful location information even in the absence of specific training sets. For example, the *Wireless Geographic Logging Engine* (http://www.wigle.net) provides pervasive mappings of wireless information to approximate geo-spatial information. It should be possible to build "federated" Mirage server that uses that information to direct users to existing location-based search engines, such as those provided by Google or Yahoo.

At the current time, the Mirage system is being developed under an "open source" development model that will allow the rapid expansion of location-based systems. The complete system will be available for download in May 2005.

REFERENCES

Bahl, P. and Padmananbhan, V. N. (2000). Radar: An in-building rf-based user location and tracking system. In *IEEE INFOCOM 2000*, Tel-Aviv, Israel. IEEE Computer Society Press.

Battiti, R., Le, N. T., and Villani, A. (2002). Location-aware computing: a neural network model for determining location in wireless lans. Technical Report Technical Report DIT-02-083, Informatica e Telecomunicazioni, University of Trento.

Byers, S. and Kormann, D. (2003). 802.11b access point mapping. In *Communications of the ACM*, volume 46, no. 5, pages 41–46.

Caswell, D. and Debate, P. (2000). Creating web representations for places. In *In Proceedings of the Second International Symposium on Handheld and Ubiquitio,us Computing (HUC 2000)*, Bristol, UK.

Cheverst, K., Davies, N., Mitchell, K., and Friday, A. (2000). Experiences of developing and deploying a context-awar tourist guide: The guide project. In *Proceedings of MOBICOM 2000*, Boston, MA. ACM Press.

Griswold, W. G., Shanahan, P., Brown, S. W., , R. B., Ratto, M., Shapiro, R. B., and Truong, T. M. (2002). Activecampus — experiments in community-oriented ubiquitous computing. In *Technical Repeort CS2003-0750*, UC San Diego. Computer Science and Engineering.

Gruteser, M. and Grunwald, D. (2003). Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis. In *Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, pages 46–55. ACM Press.

Hightower, J. and Borriello, G. (2001). Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66.

Kindberg, T., Barton, J., Morgan, J., Becker, G., Bedner, I., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Pering, C., Schettino, J., Serra, B., and Spasojevic, M. (2002). People, places, things: Web presence for the real world. *MONET*, 7(5).

MediaWiki-Foundation, T. (2004). Mediawiki development. `http://mediawiki.sourceforge.net`.

Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. (2000). The cricket location-support system. In *Proceedings of the 6th ACM MOBICOM*, Boston, MA.

Russell and Norvig (2002). *Artificial Intelligence, A modern Approach*. Prentice Hall Series in Artifical Intelligence. Prentice Hall. Chapter 20 Statistical Models (pg 733).

Schilit, B. N., LaMarca, A., Borriello, G., Griswold, W., McDonald, D., Lazowska, E., Balachandran, A., Hong, J., and Iverson, V. (2003). Challenge: Ubiquitous location-aware computing and the "place lab" initiative. In *The First ACM International Workshop on Wireless Mobile Applications and Services on WLAN (WMASH 2003)*. ACM Press.