# WLocator: An Indoor Positioning System

Shaun Phillips, Michael Katchabaw, Hanan Lutfiyya
Department of Computer Science
The University of Western Ontario
London, Ontario Canada
Email: sphill8,katchab,hanan@csd.uwo.ca

***Abstract*— There exists a multitude of location-sensing systems utilizing wireless technology, though varying in cost, coverage and accuracy. In this paper we will introduce *WLocator*, a system that aims to strike a balance between these areas. *WLocator* will provide location information relative to symbolic, user-created "landmarks". These landmarks are easily created and maintained, thus reducing short and long term costs. Additionally, the unique matching algorithm invoked by *WLocator* attempts to offer a relatively high level of accuracy, given the low cost of implementing the system.**
**Keywords: Indoor positioning systems, WiFi**

## I. Introduction

A location positioning system provides position information about an entity that is not dependent on any application. Applications that use the location position information are called *location-aware* applications. An example of a location-aware application is one that uses location information to provide directions to the user.

A key weakness of early work (e.g., [6], [7], [14]) in indoor location positioning systems is that specialized hardware needs to be installed in environments. Although the hardware itself may not always be expensive, these systems can be costly to deploy and maintain. As a result, many organizations would not be able to afford this type of solution.

The widespread adoption of WiFi in indoor environments provides an opportunity to develop low-cost, indoor positioning systems. Examples of positioning systems are found in [1], [8], [9], [19], [17], [20]. Several of these systems provide high granularity (i.e., locations that are close to each other can be distinguished) but often incur a high cost of deployment since a thorough mapping of the environment is required in advance. This mapping involves the collection of signal strengths from multiple preselected positions to create a radio map. Signal strengths from an unknown position are compared to the signal strengths of the preselected locations (found in the radio map) to determine the location of the unknown position. Thus a change in the set of access points requires that the radio map be re-created. Signal strengths measured from a location often vary since the propagation of these signals is affected by reflection, defraction and scattering of radio waves caused by frequent movement of objects, people and interference from competing signals. Several of the positioning systems handle this by taking multiple samples and applying functions. The problem with much of this work is that it makes the off-line analysis time consuming and does not address temporal variations of signal strengths. The samples are often taken within a short period time e.g., over a five minute period, or at two different times of the day. The amount of variation is reported to be around 10 dBM in the literature, while our own site surveys [13] taken over multiple days at various building at the University of Western Ontario suggests a mean fluctuation of 21 dBM.

These weaknesses led to other systems [5], [15] that provide lower granularity but do not have a high cost of deployment. The Herecast [15] location positioning system shows that even with the lower granularity that there are still interesting location-aware applications that can be developed.

This paper describes an indoor positioning system that is simply called *Western Locator* or *WLocator* that is primarily based on WiFi and addresses some of the weaknesses found in the existing work. This includes fluctuating signal strengths and the ability to deal with changes in the set of access points without the work and costs associated with other systems. Furthermore, the *WLocator* system is able to support the use of other devices that emit detectable signals e.g., Bluetooth, RFID tags. Although neither of these types of devices are commonplace today, their usage is definitely increasing. Locator's support of these devices will be illustrated through the use of Bluetooth devices.

The outline of this paper is as follows. Section II describes related work. Section III describes the algorithms developed for for location purposes within the *WLocator* system. These algorithms accommodate signal strength fluctuations and changes to the set of access points. Section IV describes a design separating location determination from application logic, providing an application programming interface that allows any application to access and use location information, which is a critical need of any location positioning system. Section V describes a location-aware application that makes use of the API described in Section IV. Section VI describes experimental results. Section VII concludes with a discussion of the experience with *WLocator* obtained through using the application and future work.

## II. Related Work

Two categories of work have been identified in location-based positioning systems that use WiFi access points [17]. The first category of location positioning systems models the propagation of radio signals (e.g., [8], [9], [16]). The work in [8] describes calibration-free location estimation techniques. This requires a location server from which the client device requests location information. The location server requires that the client device measures and reports signal strengths from all access points that have been detected. The location server uses these measurements to generate multiple distance mapping curves for each detected access point. The curves map an signal strength value to distance. Location estimation is based on triangulation. Essentially this technique makes use of a mathematical model of propagation. Other work using mathematical models of propagation includes [9], [16]. Most of the models developed assume idealized conditions (e.g., fluctuating signal strengths) that are rarely satisfied in practice.

The second category uses empirical data to approximate the location (e.g., [1], [2], [17], [19], [20]). An early example of a location positioning system that uses empirical data is RADAR [2]. In RADAR, an area is divided into a $1 \times 1$ meter grid, with signal strength measurements of access points taken at each intersection. The mean of these measurements is recorded with a timestamp using a driver on a mobile host which extracts the signal strength from the network interface card. This recording is referred to as a *signal strength tuple*. This is used to create a radio map which is a set of signal strength tuples. The creation of the radio map takes place in an off-line analysis phase. This map is used for on-line analysis where observed signal strengths are compared with entries in the radio map to determine the location whose associated signal strength best matches the observed signal strengths. The K-nearest neighbor approach is used to find the best match. Privacy is an issue since the location of a user is always known to the location system infrastructure.

The Horus system [19] is similar to RADAR in that it uses sampled signal strengths from various locations. However, instead of using a mean of signal strength values, a signal strength probability function is created i.e., a function that calculates the probability of the signal strength of a access point at location (x,y). The radio map stores the distribution of signal strength received from each access point for each sampled position. An algorithm is presented that takes as input a sequence of observations from each access point. The sequence is ordered by signal strengths. For the first access point the probability of each location in the radio map set being the correct location is calculated based on the first access point. This provides a set of possible locations based on this access point. If the probability of a location is considered high enough then this is the chosen location. A extension of Horus can be found in [17].

RADAR and Horus are limited in that the user's location can only be mapped to one of the locations defined in the radio map. The work in [1], [20] attempt to address this issue using interpolation techniques for location estimation.

Results vary with the latest results reported in [17] of being able to distinguish locations just over four meters apart. The Horus system [19] is suppose to be able to distinguish locations about 1.4 meters apart with 90%. On the other hand, [1] reports that in a performance comparison Horus has 40% accuracy while Radar has 15% accuracy. Generally we found that comparisons are difficult since different systems are tested in different environments.

As stated in Section I, signal strengths measured from a location often vary. Several of the positioning systems handle this by taking multiple samples and applying functions. The problem with much of this work is that it does not fully handle temporal variations.

Recently two location positioning systems [5], [15] using an empirical approach were developed that do not require initial calibration and changes in the set of access points do not have the maintenance costs associated with many of the systems discussed earlier in this section. Placelab [5] and Herecast [15] allow a WiFi-enabled client device to automatically determine its location by listening for signals from known 802.11 access points in the environment. Access points are known since an access point broadcasts its MAC address. Placelab stores the MAC address, latitude and longitude of each access point in the client device before the client device is used. For each access point that the client device receives a signal for, the location is calculated as the average of retrieved latitudes and longitudes. Herecast stores the access points and the symbolic name of the location associated with the access point. The location of the client device is the one associated with the access point with the strongest signal strength. Access points with weaker signal strengths have almost no impact on the determination of the location unless these access points are the only ones detected. An access point that is removed will not generate an access point and thus does not factor into any calculations of location. Fluctuating signals in Herecast imply that different access points may have the strongest signal strength measured by the client device at different points of time. However, experiments show that these access points are often close to each other and thus the inferred location is usually reasonable. PlaceLab computes location based on latitude and longitude. Fluctuations in signal strength may result in access points that are far away from the client device to not be considered in the average. Additional access points are handled by periodically downloading an updated list of access point information (e.g., latitude, longitude).

Placelab is intended for primarily outdoor use where there is a high density of WiFi access points as the result of buildings being close to each other. Herecast is intended for indoor use. The closest access point approach of Herecast, and the triangulation techniques of Placelab offer accuracies in the range of 20-25 meters. In both cases the accuracy depends on the density of access points. This level of accuracy limits possible applications. If an application requires a finer granularity than Herecast and PlaceLab can provide, then a different approach must be used.

Our work is empirical, based on our earlier work with Herecast, except that new algorithms and functions have been developed to deal with a change to access points by applying functions in the on-line analysis phase. This work also shows that other devices can be used.

## III. Algorithms

The location positioning system introduced in this paper functions by mapping *fingerprints* of received signals at a given location to fingerprints for known locations or landmarks within the environment. To more precisely explain this, we introduce the following notation and algorithms. Let $L = \{l_0, l_1, .., l_{n-1}\}$ represent preselected positions. It is assumed that each position, $l_i$ is chosen to represent a landmark i.e., a location that is easily identifiable using symbolic names. Let $A = \{a_0, a_1, .., a_{n-1}\}$ represent the wireless access points in a building. Let $B = \{b_0, b_1, .., b_{m-1}\}$ represent the set of Bluetooth devices. For any location, $l_i$, the set $\{r_{i0}, r_{i1}, ...r_{in-1}\}$ is the set of signal strengths observed for each device in $A$. A similar set can be defined for Bluetooth devices except that instead of recording signal strengths, one value is assigned for those access points that have been detected and another value is assigned for those access points that have not been detected (more on this later in this section).

A fingerprint is collected for each preselected position in $L$ and stored. For an unknown position, $x$, the fingerprint from a preselected position that best matches the fingerprint taken for $x$ determines the location. (Note that in the rest of the paper $l_i$ is being used to denote a fingerprint associated with a specific preselected position, $i$.)

This remainder of this section describes algorithms for the following: (i) Finding the best match between the fingerprint taken at the current position as measured by the client device and the set of fingerprints associated with the preselected locations (section III.A); (ii) Algorithms for updating fingerprints of preselected locations (section III.B).

### A. Matching Algorithm

This work describes an algorithm that finds the best match between the fingerprint taken of the current position as measured by the client device with the signal sets associated with the predetermined locations. The matching algorithm takes as input a fingerprint of the current location of the client device, $x$, and for each fingerprint in $L$ computes a similarity measure that is used to compare $x$ with the fingerprints of predetermined positions represented in $L$. A device not detected is represented by the constant $Low$. The value of $Low$ is a low negative number since the range of values for WiFi access point signal strengths includes negative numbers.

```
Input: $L$: Set of fingerprints for predetermined
       positions
Input: $x$: Fingerprint associated with current
       unknown position
Input: $D$: Set of WIFI and Bluetooth devices
Output: $SimilarityMeasures$: Set of similarity
        measures for each $l \in L$
1  $sum_x$ = 0
2  foreach $d \in D$ do
3      if $r_{xd} > Low$ then
4          $sum_x$ = $sum_x$ + $score(r_{xd})$
5      end
6  end
7  foreach $l \in L$ do
8      foreach $d \in D$ do
9          if $r_{ld} > Low$ then
10             $t_1$ = $score(r_{ld})$
11             $t_2$ = $adjust(r_{xd}, r_{ld})$*$t_1$
12             $sum_l$ = $sum_l$ + $t_2$
13         end
14         $ratio_l$ = $sum_l$/$sum_x$
15         $similarity_l = \frac{(similarity_l^p * weight + ratio_l)}{(weight+1)}$
16         Add $similarity_l$ to
           $SimilarityMeasures$
17     end
18 end
```
**Algorithm 1**: Matching Algorithm

Lines 1-4 of the algorithm show the computation of the sum of signal strengths after the *score* function has been applied for fingerprint $x$ resulting in $sum_x$. A similar computation, with adjustment made for signal fluctuations (using the *adjust* function), is done for each $l \in L$ (Lines 8-13) resulting in $sum_l$. The upper bound of this value is $sum_x$. Higher values of $sum_l/sum_x$ indicate better matches (Lines 12-15). This is referred to as the *similarity measure*. The similarity measure needs to compensate for fluctuating signal strengths, added/dropped access points and Bluetooth devices. The two functions, *score* and *adjust* are used for this purpose.

The following is a description of how *score* and *adjust* behave when the device is a WiFi access point. This will be followed by a brief discussion of how these functions behave when the device is a Bluetooth device. The *score* function is linear as graphically depicted in Figure 1. This function is applied to the signal strength measured for each detected access point (line 10). The higher the signal strength of an access point the higher the value returned by *score* to be

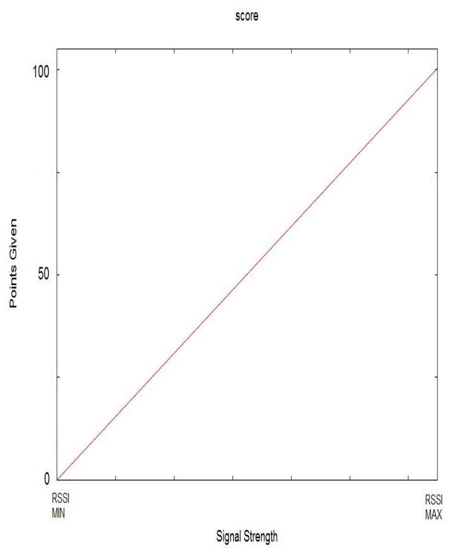


Fig. 1. The Score Function

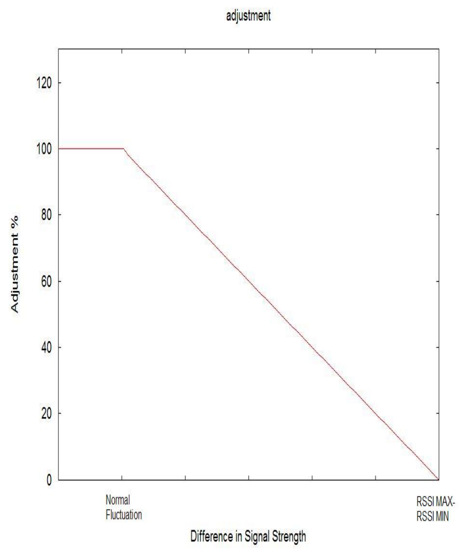


Fig. 2. The Adjust Function

associated with that access point. For a position, an access point with a weak signal at time $t$ may not be detected at time $t'$. This should have less impact in determining the location. This is illustrated with the following example. Assume that $x$ and $x'$ are two different fingerprints of the same unknown position taken at different points of time ($x$ is taken before $x'$) and that $x$ reflects that it detects a signal strength from WiFi access point $k$ not detected by $x'$ i.e., $r_{xk}$ is greater than $Low$ but $r_{x'k}$ is $Low$ suggesting that $k$ is a relatively far away access point. $score(r_{xk})$ results in a low number since the signal is weak while $score(r_{yk})$ is zero. $score(r_{xk})$ is not much larger than $score(r_{yk})$ and thus has relatively little impact on the value of $sum_l$. The similarity measures computed will not change very much implying that the location computed will be the same for $x$ and $x'$ unless two preselected locations have similar fingerprints. In this case, the access point with the weak signal may be the deciding factor.

Not all signal strength fluctuations are associated with access points with weak signal strengths. In addressing the issue of fluctuating signal strengths, the matching algorithm uses the *adjust* method. Similar to *score*, *adjust* is modelled as graphically depicted in Figure 2. For a range of differences in signal strength to a point, *adjust* function returns 100%. This is referred to as the *fluctuation threshold* point. It then decreases linearly.

The difference in signal strength for a WiFi access point is calculated as the difference in dBm between an access point that is common between the fingerprints being compared. The function returns the percentage by which to adjust the value given by the *score* function (lines 10, 11). Given that signal strength fluctuations are commonplace, the *adjust* function accounts for this by disregarding fluctuations within a certain range. Thus, any difference in signal strength that is within the range of possible fluctuations implies that the *score* function is applied fully. As an example, data collected from a random sampling of fingerprints in the Middlesex College building at Western has shown the mean fluctuation of received signal strengths to be roughly 21dBm in this area. This suggests that for any unknown position different fingerprints taken over-time may show great variation in the signal strengths measured for access points. This suggests that if the difference between the recorded signal strength for a particular access point for $x$ and for a fingerprint of a pre-determined position is less than or equal to a threshold value, then the value returned by *score* for the access point is unaffected. However, differences greater than the threshold value will progressively diminish the value of the access point, as these may indicate that the fingerprints being compared represent are unique locations.

As noted earlier Bluetooth devices are either detected or not detected since current technology does not easily allow for signal strength measurement as can be done for WiFi access points. For Bluetooth devices using Boolean values, zero (indicating that a device has not bee detected) and one (indicating that a device has been detected), cannot be used since a value of one would not contribute greatly to the sum being computed and hence would make it more difficult to distinguish fingerprints. Thus for a Bluetooth device if a device is not detected then it is assigned the value *Low* otherwise it is assigned a fixed number as the signal strength. The score function is applied to these values. The *adjust* function returns an adjustment percentage of 100% if the device is detected otherwise it returns zero. It is assumed that the *score* and *adjust* functions know the device type (e.g., Bluetooth or WiFi access point) that the function is being applied to.

The final calculation takes place in line 15. For every comparison of an unknown position a similarity measure is computed for each preselected location, $l_i$. The last similarity measure is denoted by $similarity_l^p$. The *weight* indicates the emphasis that should be placed on the current measurement. For example, a weight of two will place 50%, a weight of four will place 25%, etc. A low weight forces fingerprint values to change quickly as more emphasis is placed on the current measurement as opposed to previous measurements. A high weight causes fingerprint values to to change more slowly as more emphasis is placed on the previous measurement as opposed to the current measurement. The reason for this step is to deal with variations of signal strength caused by small user movement of the client device.

### B. Updating Fingerprints of Predetermined Locations

The addition and deletion of access points or Bluetooth devices requires the use of new fingerprints for preselected positions.

For a preselected position let $l$ be the current fingerprint and $l'$ represents the new fingerprint. For each $d \in D$ the value of $l'_d$ is computed as follows:

$$l'_d = \frac{(l_d * weight + l_d)}{(weight + 1)}$$

If the added device is a WiFi access point then the added access point's RSSI is averaged with the minimum RSSI of the client device since it did not exist in $l$. Weights are used in the same fashion as discussed in the previous subsection. The following example is used to illustrate how fingerprints of preselected locations are updated. Assume that $D = \{b_0, b_1, b_2\}$ where $b_2$ represents the latest access point to be added. Let $l$ = {12,20}. Let $weight$ be equal to 0.25. The signal strength of the added access point is averaged with the client device's minimum RSSI value. This means that $l'$ = {12,20,-18}. Deletion of access points also uses this technique. However, upon a second reading if an access point no longer exists then it is taken out of the fingerprint. This allows applications to recalibrate the system without having to update all the fingerprints at once. Thus on-line analysis can still take place (with perhaps lower accuracy) while recalibration takes place.

Since Bluetooth devices do not have fluctuating signal strengths newly added Bluetooth device results in a variable in the fingerprint whose value reflects that a new Bluetooth device has been detected.

## IV. Design

This section briefly describes the design of the *WLocator* system. More details of the API and design can be found in [13].

### A. Scanner

An object of the Scanner class is responsible for monitoring wireless signals. Only one such object is allowed within an application. The Scanner object coordinates objects of other classes (helpers) where a class provides the functionality needed to monitor a specific type of device e.g., wireless access point, Bluetooth devices. Each helper class maintains a list of the detected devices and the signal strengths of each of the detected devices. These lists can be provided to requesting classes. This work used two helper classes: (i) The WiFiScanner class is used for wireless access points; (ii) The BluetoothScanner class is used for Bluetooth devices. Currently detected devices are maintained in a list that is returned to the Scanner when requested by the Scanner.

Only Bluetooth devices that would be considered to be stationary are tracked, as including potentially mobile devices would be detrimental to accuracy. For example, if a fingerprint can reflect the use of a cellular phone then the created fingerprint will consist of data that is almost certainly unreliable since cellular phones are mobile and the device will unlikely be in the fingerprint at subsequent readings. Keyboards, and mice are examples of relatively stationary Bluetooth devices.

After a Scanner object is created the developer has the option of creating fingerprints using the `getFingerprint` method. After receiving a request for a fingerprint using the `getFingerprint` method, the Scanner obtains a listing of currently detected Bluetooth devices and wireless access points from the helper classes. This is done through the use of two methods that each helper class has. The `BeginScan` method starts a scan of the environment that the client device is in for signals for the devices that it is associated with. The `EndScan` method terminates the scanning initiated by a call to `BeginScan`. The last signals detected for the WiFi and Bluetooh devices are provided to the Scanner object and are maintained until the Scanner object is destroyed.

### B. Fingerprint

The *WLocator* system associates a symbolic name with each fingerprint (similar to that found in [15]). Symbolic names have more meaning and are more readily identifiable than their numerical counterparts. For example, an application would be more engaging to a user if the user was informed that the location is "Bill's cubicle" rather than "-100.345, 32.871." A class is used to encapsulate the storage and retrieval of fingerprints and symbolic names.

### C. Manager

The Manager class provides facilities for developers to perform various actions on fingerprints. Over the lifetime of a location-aware application, potentially thousands of fingerprints can be created and used.

The application has a choice when creating a Manager object on providing a file name with fingerprint data. This data represents the fingerprints of predetermined locations. The `WriteData` method can be used by a location-aware application to write fingerprint information to a file. There is a corresponding method for reading fingerprint data.

The Manager class is able to compare fingerprints by computing a similarity number. This functionality is used to determine which stored fingerprint most accurately describes the current location. This was described in the previous section. A method of the Manager class, `getTopFingerprints`, is used by location-aware applications to get the symbolic names of the fingerprints of preselected positions that most closely match (using the algorithm in III.A) the fingerprint of the current location. The number of symbolic names to be returned is given as input.

Another method of the Manager class is the `merge` method. This implements the algorithm described in Section III.B. The inputs are the fingerprint of the current location measured by the client device and the index of the fingerprint in storage to be updated.

The `updateLocations` method takes as input the fingerprint of the current position measured by the client device and determines the location based on the algorithm described in III.A. Each preselected position is assigned a new similarity measure.

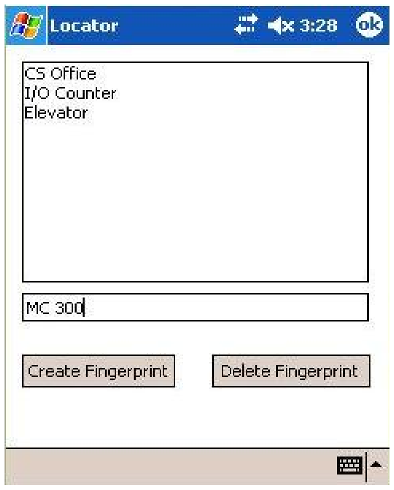
## V. Implementation

The client software is written in embedded Visual C++ for Windows Mobile/PocketPC environments. This was put on an HP iPAQ 5500. The file that fingerprint data is written out to is textual and local to the PDA. The WiFi Scanner open source software [18] is used to the interface to the embedded network card to discover access points.

The WiFiScanner object maintains a list of all detected wireless access points and their signal strengths. The WiFiScanner class provides a method to retrieve the list of detected wireless access. Fingerprints are created by using the signal strengths of devices detected in the list of all detected wireless access points while all other access points have *Low* recorded as the signal strength.

This raises the question of how long an access point should remain in the list of detected wireless access points. *WLocator* allows for 10 seconds of not detecting a signal from an access point before it is purged from the list. Certainly, a lower threshold would allow the system to be more responsive to change, however, testing has shown that a period of 10-15 seconds provides an ideal mix of responsiveness and reliability. That is, a weak access point will occasionally not be on the list of detected access points and a grace period of at least 10 seconds ensures that the access point remains in the list of detected access points when the user's location is stable.

Scanning for Bluetooth devices is similar to scanning for wireless access points. However, difficulties arise because there exists several implementations of the Bluetooth protocol stack. For example, both Broadcom (formally WIDCOMM) and Microsoft offer different implementations, and there are at least two more



CS Office
I/O Counter
Elevator

MC 300

Fig. 3. Fingerprint Management

for the Linux platform. As a consequence, code is not portable between devices that use different Bluetooth stacks. This work used the Broadcom stack.

Our experiments on campus suggested that even though it is possible for there to be a fluctuation of 21dBm this is too large for a fluctuation threshold since this does not occur often. Observed signal strength fluctuations are the result of several factors including hardware of the source device, interfering objects and signals, and radio wave propagation. Consequently, the level of fluctuation is dependent on the environment. Thus, the threshold should be set to a value that will be applicable to as many environments as possible in which the system will be used. By default, *WLocator* initializes the threshold value to 5dBm. The API provides a method that allows this value to be adjusted by the user or developer.

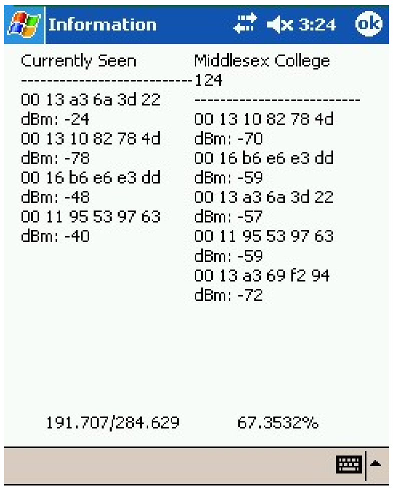
## VI. Location Application

This section describes a sample application created using the implementation based on the design and algorithms defined in Sections III and IV. This application creates and manages fingerprints. It was used for the experiments described in Section VII.

The start of the program causes the creation and initialization of a Scanner object allowing the application to create fingerprints using the `GetFingerprint` method (see Figure 3).

Created and stored fingerprints may be displayed at any time by clicking on the names in the list shown in Figure 4. A separate dialog is opened and the selected fingerprint is displayed along with what is currently detected by the system. This is shown in Figure 5.

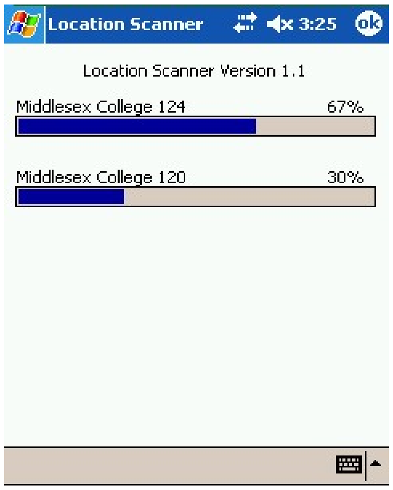
Using a timer, the application uses continuous calls to `updateLocations` to determine the location of the current position by comparing it with the fingerprints of the preselected position. The `GetTopFingerprints` methods returns the fingerprints with the highest similarities to be displayed. This is graphically depicted in Figure 5.



```
Currently Seen              Middlesex College
----------------------------124
00 13 a3 6a 3d 22           ---------------------------
dBm: -24                    00 13 10 82 78 4d
00 13 10 82 78 4d           dBm: -70
dBm: -78                    00 16 b6 e6 e3 dd
00 16 b6 e6 e3 dd           dBm: -59
dBm: -48                    00 13 a3 6a 3d 22
00 11 95 53 97 63           dBm: -57
dBm: -40                    00 11 95 53 97 63
                            dBm: -59
                            00 13 a3 69 f2 94
                            dBm: -72


  191.707/284.629          67.3532%
```

Fig. 4. Display of fingerprints



```
Location Scanner Version 1.1
Middlesex College 124                 67%
Middlesex College 120                 30%
```

Fig. 5. Display of fingerprints

The development of this application does not require the developer to know how to scan for fingerprints or how location is actually determined. The developer was able to focus on the development of the GUI. This suggests that the design and its interface does allow for the separation of location determination from using the location information, which is a key benefit to using *WLocator*.

## VII. Experimental Evaluation

This section describes the evaluation of the system. Sections A-D focus on the results of experiments where it was assumed that only WiFi access points are used, while Section E discusses experiences from experiments involving the use of Bluetooth devices as well.

### A. Experimental Testbeds

Experimentation took place in three different testbeds. The first testbed is in the basement, second and third floors of the Middlesex College building. The second testbed is the first and second floor of

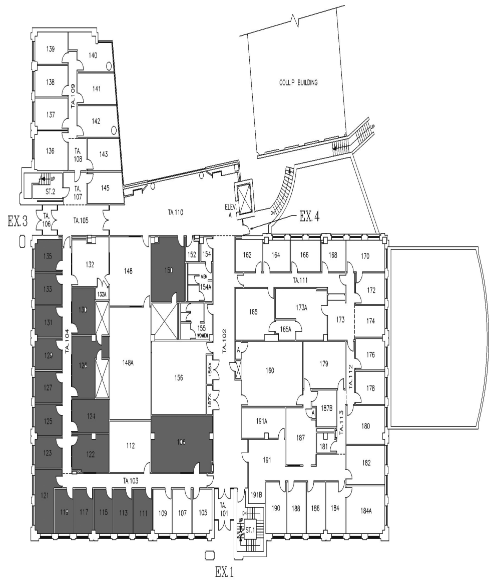


Fig. 6. Western Science

the Western Science building (graphically depicted in Figure 6). The third testbed is the ground floor of the Taylor library in the Natural Sciences building (graphically depicted in Figure 7). All three buildings are at the University of Western Ontario. The buildings vary in terms of the size, density of access points and number of walls. We do not have control over the placement or number of access points. This approach to choosing testbeds reflects a realistic environment that lessens the likelihood that adjustments to parameters or algorithms are the result of the characteristics of a very specific environment. The Middlesex College building and Western Science buildings have 11 access points each. However, the Western Science building is a smaller building and thus has a higher density of access points. The typical number of WiFi access points that could be seen from a location ranged from three to seven.

### B. Data Collection

For each building fingerprints were created three, five and ten meters apart. Between 20 and 30 fingerprints of predetermined locations were taken on each floor. The preselected positions were primarily in the corridors and computer laboratories since we do not have access to most offices.

### C. Summary of Performance Results

Analysis of the experimental results shows that the *WLocator* system had a granularity of 5-10 meters. The system could differentiate between locations 5 meters apart 65-75% of the time. Locations that were 10 meters apart were correctly identified 85-95% of the time. More results can be found in [13].

### D. Analysis of Results

The variation of success of *WLocator* is the result of several factors. One factor is the density of access

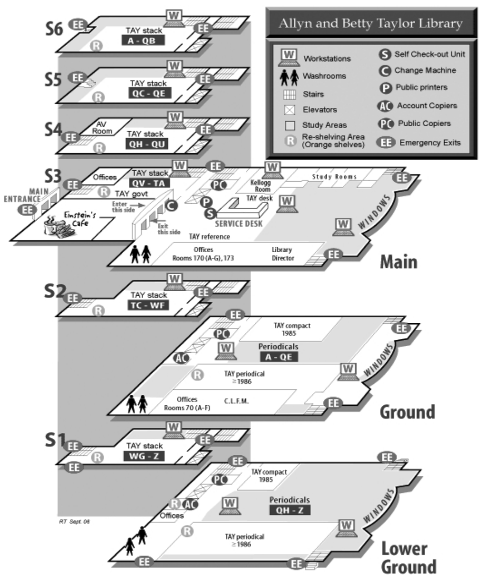


Fig. 7. Taylor Library

points within an area. If only a small number of access points service an area, it is difficult to accurately identify locations. For example, the western-most wing of the ground floor in Middlesex College is serviced by two access points. If the fingerprints are relatively close together (within, say, 20 meters), then it is difficult to identify a Fingerprint as the actual location with any certainty. Regular signal strengths fluctuations and the uniformity of the fingerprints makes determining location in under these circumstances difficult.

Accuracy is increased when fingerprints have a degree of uniqueness and there is variety amongst the fingerprints in an area. This occurs when an environment has many WiFi access points that can be detected at any position. The first floor of the Western Science building is an example of an environment where this situation occurs. Fingerprints in this area show that at least 10 wireless access points with signal strengths can be detected. This allows the algorithm defined in III.A a good deal of data to make comparison. This is seen in the results which show the highest accuracies.

Another factor are the obstacles to radio wave propagation. We observed that its presence can aid in determining location. We have discussed the extreme scenarios where environments have either very few devices, or an abundance of devices. Usually the number of WiFi access points varied from three to seven. It was found that two fingerprints taken five meters apart with nothing to separate the fingerprints are difficult to distinguish. If a wall is added between them, a fingerprint may lose sight of an access point which has a weaker signal. The matching algorithm can better distinguish the fingerprints since there is now a weaker signal. This phenomena was observed in buildings that had many corridors and offices that were compacted together in a small space–such as the first floor of the Western Science Center (see figure 6). Conversely, the wide open spaces of the ground floor of Taylor library (see figure 7) ensured that fingerprints remained relatively similar and difficult to distinguish.

### *E. Use of Bluetooth Devices*

Experimentation was limited due to a lack of Bluetooth devices. However, from what was observed, the use of Bluetooth devices could potentially be very useful. Bluetooth devices in the environment proved to be particularly beneficial to the accuracy of the system. High-powered Bluetooth devices, which can broadcast their signals up to 100 meters, generally act as an extra wireless access point (and more is always better for location determination). However, the most useful and common devices are low-powered and transmit their signal up to ten meters. When this is the case there are relatively few fingerprints that show that this device has been detected. If there was only one fingerprint that saw the Bluetooth device then the accuracy of results was close to 100%. Having Bluetooth devices show up as detected in only one fingerprint is more likely to be found in an environment such as the Western Science building. The presence of low-powered Bluetooth devices adds greater variety to fingerprints in a close range. If a low-powered Bluetooth device is typically seen by only a single fingerprint then the matching algorithm can easily identify the fingerprint by the detection of the Bluetooth device. In our experiments the accuracy was increased by 10%.

## VIII. Conclusions

This work described the *WLocator* indoor positioning system. This system is different from others proposed in the literature in several ways as discussed below. This section briefly describes observations and future work.

**Distinction between Off-line and On-line Phases**. Most of the systems described in Section II that use empirical data have distinct off-line and on-line phases. The off-line phase is used to build a radio map that can be used in the on-line phase to determine the location of an unknown position. These phases do not overlap. On the other hand, *WLocator* allows for these phases to overlap as described in section III.B. This makes the task of recalibration easier.

**Dealing with Fluctuations**. The *score* and *adjust* functions, defined in Section III.A, deal with fluctuations by addressing the causes of fluctuations. One source of fluctuations are access points that are far away and thus cannot be detected or provide weak signals. The *score* function ensures that these do not contribute heavily to determining the location of the unknown position. The *adjust* deal with the random fluctuations that often occur even when the access point may be close by. Although there exists some functionality in other systems to deal with fluctuations we chose to develop and experiment with simpler functions. The reason is that the functions used in other work are used in an off-line phase. We need functions that work well in an on-line phase and thus must not be computationally intense. Future work will look at different implementations of the *score* and *adjust* functions. For example, the *adjust* function currently uses a constant and then linearly decreases. A step function may be more appropriate.

**Changing Set of Devices**. Section III.B shows a technique for updating fingerprints when the set of access points are updated. Most of the other location positioning systems do not address this problem.

**Device Changes**. A problem with other systems is that a change in devices distorts the results. The approach used in this paper is more resilient to differences between devices.

**Fingerprint Management**. The set of fingerprints of preselected positions is stored in a file. This file can be downloaded by a location-aware application. The location-aware application can use the API to create a Manager object which has passed to it the name of the file. The Manager object uses this as the set of fingerprints of preselected positions.

**Propagating Changes of Fingerprints of Preselected Positions**. There has been little discussion about how client devices get fingerprints of preselected positions. Clients can download a file with fingerprints of these positions. Updates of this file can be done periodically based. There are a couple of approaches that can be used to update this file. One approach would have an administrator update the file when an access point is added. Alternatively there could be a community driven effort for recording fingerprints which would be similar to Herecast [15]. This will be incorporated into future versions of *WLocator*.

**Location-Aware Application Development**. The application developed was intended for experimental purposes. It was found relatively easy to develop in that its development did not require knowledge of location determination. However, more experience is needed in the development of applications to determine if the API is sufficient or if additional methods are needed.

**Lightweight**. The software was placed on a HP PDA. This is unlike much of the other work in this area that did testing and experimentation on laptops. This work uses a PDA illustrating that it is lightweight. Future work will deploy the software to a WiFi-enabled cell phone.

**Performance**. In general, it is difficult to estimate the performance of *WLocator* in a new environment. The network topology and building layout greatly effect the granularity of the system. The testing completed at the University demonstrates how these factors ultimately aid or hinder the system's accuracy. However, across all test environments, the system could reliably identify locations that were five to ten meters apart. The degree of certainty varied slightly, and was often the result of the aforementioned factors. In particular, a large pool of devices that allowed for unique fingerprints, and building layouts that provided obstacles to radio wave propagation created environments where *WLocator* achieved its sought granularity. We plan to continue deploying *WLocator* to other buildings on campus and increase our usage of Bluetooth devices.

## REFERENCES

[1] A. Agiwal, P. Khandpur and H. Saran, "Locator: Location Estimation System for Wireless LANs," Proceedings of the 2nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots, 2004, pp. 102-109.

[2] P. Bahl and V. Padmanabhan, "Radar: An in-building RF-based user location and tracking system", *IEEE Infocomm 2000*, vol. 2, pp. 775-784.

[3] Joe Bardwell, "Converting Signal Strength Percentage to dBm Values," 2002.

[4] Mike Hazas, James Scott, and John Krumm, "Location-Aware Computing Comes of Age," *IEEE Computer*, Vol. 37, No. 2.

[5] Gaetano Borriello, Matthew Chalmers, Anthony LaMarca, and Paddy Nixon, "Delivering Real-World Ubiquitous Location Systems," *Communications of the ACM*.

[6] A. Harter and A. Hopper, "A Distributed Location System for the Active Office,", IEEE Network Special Issue on Distributed Systems for Telecommunications, vol. 8, no 1, 1994, pp. 62-70.

[7] A. Hopper, P. Steggles, A. Ward, P. Webster, "The Anatomy of a Context-Aware Application", Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom '99), 1999.

[8] Y. Gwon, R. Jain, "Error Characteristics and Calibration-Free Techniques for Wireless LAN-based Location Estimation," ACM Proceedings of the Second International Workshop Mobility Management and Wireless Access Protocols, 2004, pp. 2-9.

[9] P. Krishnan, A. Krishnakumar, H. Ju, W. Ju, C. Mallows, and S. Ganu, "A System for Lease: Location Estimation by Stationary Emitters for Indoor RF Wireless Networks", INFOCOM 2004.

[10] R. Jarvensivu, R. Pitkanen and T. Mikkonen, "Object-Oriented Middleware for Location-Aware Systems,", ACM Proceedings of Applied Computing, 2004, pp. 1184-1190.

[11] Jeffrey Hightower, Sunny Consolvo, Anthony LaMarca, Ian Smith, and Jeff Hughes, "Learning and Recognizing the Places We Go," *In Proceedings of Ubicomp*, 2005, Tokyo, Japan. September 2005.

[12] J. Hightower, G. Borriello, "Location Systems for Ubiquitous Computing," *IEEE Computer*, 2001, pp. 57-66.

[13] S. Phillips, "Locater: An Indoor Positioning System", Undergraduate thesis report, 2007.

[14] N. Priyantha, A. Chand and Y. Lee, "The Cricket Support System," MOBICOM, 2000, pp. 32-43.

[15] Mark Paciga, "Herecast: An Open Infrastructure for Location-Based Services using Wi-Fi," M.S. thesis, University of Western Ontario, London, ON, Canada, 2004.

[16] A. Smailagic, D. Siewiorek, J. Analt, D. Kogan, and Y. Wang, "Location Sensing and Privacy in a Context Aware Computing Environment," Pervasive Computing, 2001.

[17] J. Wierenga, P. Komisarczuk,"SIMPLE-Developing a LBS Positioning Systems," ACM Proceedings of the 4th International Conference on Mobile and Ubiquitous Multimedia, 2005, pp. 48-55.

[18] "WiFiScanner: A New 892.11b Scanner," http://wifiscanner.sourceforge.net

[19] M. Youssef, A. Agrawala and U. Shankar,"WLAN Location Determination via Clustering and Probability Distributions," IEEE PerCom 2003.

[20] Z. Xiang, S. Song, J. Chen, H. Wang, J. Huang and X. Gao, "A Wireless LAN-Based Indorr Positioning Technology," IBM Journal of Research and Development 48, 5-6, 2006, pp. 617-626.