UbiComp 2004. The Sixth International Conference on Ubiquitous Computing, pp. 283-300. September 7-10, 2004, Nottingham, England.

The NearMe Wireless Proximity Server

John Krumm and Ken Hinckley

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA, USA

jckrumm@microsoft.com
kenh@microsoft.com

Abstract. NearMe is a server, algorithms, and application programming interfaces (APIs) for clients equipped with 802.11 wireless networking (Wi-Fi) to compute lists of people and things that are physically nearby. NearMe compares clients' lists of Wi-Fi access points and signal strengths to compute the proximity of devices to one another. Traditional location sensing systems compute and compare absolute locations, which requires extensive *a priori* calibration and configuration. Because we base NearMe entirely on proximity information, NearMe works "out of the box" with no calibration and minimal setup. Many "location-aware" applications only require proximity information, and not absolute location: examples include discovering nearby resources, sending an email to other persons who are nearby, or detecting synchronous user operations between mobile devices. As more people use the system, NearMe grows in both the number of places that can be found (*e.g.* printers and conference rooms) and in the physical range over which other people and places can be found. This paper describes our algorithms and infrastructure for proximity sensing, as well as some of the clients we have implemented for various applications.

1 Introduction

One of the goals of ubiquitous computing is to build applications that are sensitive to the user's context. An important part of context is the list of people and places that are close to the user. One common way to determine proximity is to measure absolute locations and compute distances. However, computing absolute location is not necessarily easy (see [1] for a survey), especially indoors, where GPS does not work, and where people spend most of their time. The NearMe wireless proximity server dispenses with the traditional computation of absolute locations, and instead estimates proximity (distance) directly. The advantage of using proximity is that, unlike location sensing techniques, it does not require any *a priori* geometric calibration of the environment where the system is to be used.

NearMe is a server, algorithms, and application programming interfaces (APIs) meant to compute lists of nearby people and places for clients running on various 802.11 Wi-Fi devices. NearMe determines proximity by comparing lists of Wi-Fi access points (APs) and signal strengths from clients. We refer to these lists as "Wi-Fi signatures." By comparing Wi-Fi signatures directly, NearMe skips the intermediate step of computing absolute location, which means it works without calibration for clients equipped with Wi-Fi devices. Our system exploits the growing ubiquity of Wi-Fi access points, using them not necessarily as entry points to the network, but as signatures that distinguish one location from another, much like most Wi-Fi location efforts (*e.g.* RADAR[2] and Place Lab[3]).

NearMe computes proximity as opposed to absolute location. While proximity can be easily computed from absolute location, NearMe demonstrates that computing proximity directly can be much easier. Proximity is useful for polling for nearby people and places and for computing how far away they are. Proximity cannot, in general, answer questions about the absolute location of something nor how to get there. Therefore, our system is not intended to be used to find lost things nor to map routes to destinations. Instead, NearMe is intended to discover what is already nearby and to augment context for ubiquitous computing.

NearMe divides proximity into two types: short range and long range. People and places in short range proximity are defined as those with at least one Wi-Fi access point in common. We have developed a function that estimates the distance between clients in short range proximity based on similarities in their respective Wi-Fi signatures. Short range proximity is primarily intended for finding people and places within the coverage of one access point, which generally ranges from 30-100 meters. Long range proximity means that the two objects of interest are not within range of any one access point, but are connected by a chain of access points with overlapping coverage. The NearMe server maintains a list of overlapping access points that is automatically built from access point data that clients provide during the normal use of the NearMe server. The server periodically scans through all its stored access point data to create a topology of overlapping APs. It also examines time stamps on the data to create travel time estimates between pairs of access points. These travel times and AP "hops" are provided to clients as estimates of the nearness of people and places in long range proximity.

Both short range and long range proximity are computed from Wi-Fi signatures without any explicit calibration, meaning that deployment of NearMe is only a matter of getting people to run the software. People can use NearMe by running one of a few different clients we have written to run on a Wi-Fi-capable device. The client is operated by first registering with the system, sending a Wi-Fi signature to the server, and then querying for people and various types of objects or places nearby. Objects like printers and places like conference rooms and other resources are inserted into the database by a user physically visiting that place, registering as the object or place, and sending in a Wi-Fi signature. Once registered in this way, objects and places can be found by anyone else using the system. Traditional location-based systems use the same sort of registration of meaningful locations, only they also require an intermediate step of calibration to go from sensor measurements to absolute location. For instance, Wi-Fi based positioning systems need a signal strength map generated from

either manually measuring signal strengths or from simulating them based on measured access point locations, *e.g.* RADAR[2]. NearMe skips this geometric calibration step in favor of a collaborative process of registering useful locations by multiple users which are then shared with all users. Hence the system can gain acceptance by gradual adaptation without an onerous up-front investment to calibrate a specific environment. This also makes the system potentially more amenable to inevitable changes in Wi-Fi access points: If a Wi-Fi signature is no longer valid, users would be motivated to report a fresh Wi-Fi signature for the places that are important to them.

The next section of this paper describes related work. The NearMe client and server functions are discussed in Sections 3 and 4. Section 5 describes our experimental work to develop a robust function to estimate distance between clients in short range proximity by comparing their Wi-Fi signatures. In Section 6 we describe some of the applications we have implemented using NearMe, and we conclude in Section 7.

2 Related Work

The research described in this paper is related to several other projects and technologies in ubiquitous computing, including location sensing, proximity measurement, and device discovery.

There are many ways to automatically measure location [1], including Wi-Fi signal strengths, GPS, and active badges. Our proximity technique uses Wi-Fi signal strengths. Wi-Fi has been successfully used for computing location, starting with the RADAR system [2] and continuing with Intel Research's growing Place Lab initiative [3], among others. Some location systems require the deployment of specialized hardware in the environment, *e.g.* satellites for GPS and special receivers and/or transmitters for active badges. All of them require offline setup in the form of calibrating the region of use or mapping of base stations. NearMe is different in two significant ways: (1) it depends only on existing Wi-Fi access points; and (2) for finding nearby Wi-Fi devices, it requires no calibration or mapping. For finding nearby places, it only requires that the place has been registered once with the Wi-Fi signature from that location.

Proximity, as distinct from location, is an important part of a person's context. Schilit *et al.*[4], in an early paper on context-aware computing, define context as where you are, who you are with, and what resources are nearby. Note that the latter two of these three elements of context depend only on what is in a user's proximity, and do not require absolute location. Hightower *et al.* [5] describe how location-dependent parts of context can be derived from raw sensor measurements in a "Location Stack". An "Arragements" layer takes location inferences from multiple people and things to arrive at conclusions about proximity, among other things. NearMe jumps directly from sensor measurements (Wi-F signal strengths) to proximity arrangements without the intermediate complexities of computing locations.

Several systems provide wireless "conference devices" that are aimed at assisting conference attendees with proximity information. These are generally small wireless

devices that can be easily carried or worn, normally by people in large groups. Examples include nTAG [6], SpotMe [7], IntelliBadge [8], Conference Assistant [9], Proxy Lady[10], and Digital Assistant [11]. Among the features of these devices are their awareness of location and/or who is nearby. Some of them use base stations in the environment to measure location, while others use peer-to-peer communication to find other nearby conference devices. Except for the Conference Assistant, these are specialized hardware devices, whereas NearMe runs on any client that supports network access and Wi-Fi. In addition, NearMe needs no special infrastructure, and it gives proximity information about people and things that can be much farther away than the range of regular peer-to-peer communication by using its knowledge of adjacencies of overlapping access points.

There are well-established protocols for peer-to-peer device discovery using Bluetooth and Infrared Data Association (IrDA) [12]. Bluetooth works in the 2.4GHz RF range and discovers other Bluetooth devices by hopping through a sequence of channels looking for devices of a specified type, like PDAs or printers. NearMe clients may also search for specific types of things, including people, printers, and conference rooms. But unlike NearMe, Bluetooth cannot discover things that are along a chain of devices with overlapping coverage. Thus the discovery range of Bluetooth is limited to about 10 meters. While Bluetooth does not require a clear line of sight between devices, IrDA does, and it only works over a range of about one meter.

Detecting synchronous user operations, or shared context in sensor data, represents another related set of technologies. For example, “Smart-Its Friends” [13], synchronous gestures [14], and “Are You With Me?” [15] detect similar accelerometer readings due to shaking, bumping, or walking. In general, any synchronous user operation can be used to identify devices. For example, SyncTap [16] forms device associations by allowing a user to simultaneously press a button on two separate devices. Stitching [17] is a related technique for pen-operated devices: a user makes a connecting pen stroke that starts on the screen of one device, skips over the bezel, and ends on the screen of another device. This allows the user to perform an operation that spans a specific pair of devices, such as copying a file to another device. NearMe complements this class of techniques, because NearMe allows such systems to narrow the set of potential associations to only those devices that are actually in physical proximity. This helps resolve unintentional coincidences in sensed contexts, and it reduces the number of possible devices that need to be searched for association. Section 6.3 describes how we use NearMe to implement this functionality for the Stitching technique.

NearMe is most closely related to two commercial systems: Trepia [18] and peer-to-peer systems like Apple’s “iChat AV” [19]. Trepia lets users communicate with other nearby users that it finds automatically. Users can manually specify their location and Trepia also uses wired and Wi-Fi network commonality to infer proximity. While NearMe also uses Wi-Fi, it makes use of signal strengths to estimate fine-grained proximity, and it also uses an automatically updated table of physically adjacent access points to determine longer range proximity. iChat AV lets users on the same local network find each other for instant messaging or video conferencing. Similar systems for computer games let users on the same network find other nearby gamers. NearMe is more general in that it does not require users to be on the same net-

work in order to find each other, and that it lets users find nearby places as well as other people.

3 The NearMe Client

The client portion of NearMe is a program that users run to interact with the proximity server. The programmatic interface to the server is a web service which presents a simple set of APIs for a client to use, making it is easy to write new clients. We have written seven: four for Windows XP, two for Pocket PC 2003, and one in the form of an active server page (ASP). Each client performs the same three functions:

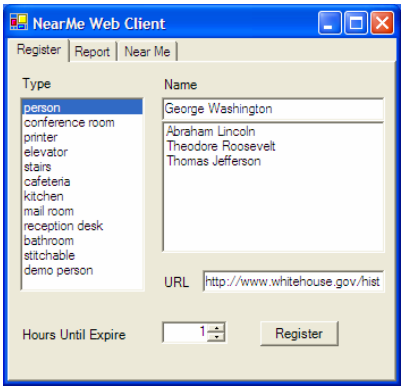
1. Register with the proximity server.
2. Report Wi-Fi signature.
3. Query for nearby people and places.

We will present a general Windows client as an example as it demonstrates most of the system's functionality. Some of the other application-specific clients are detailed in Section 6. The main work of NearMe is performed by the server, which we discuss in Section 4. The next three subsections explain the above three steps of using the client.

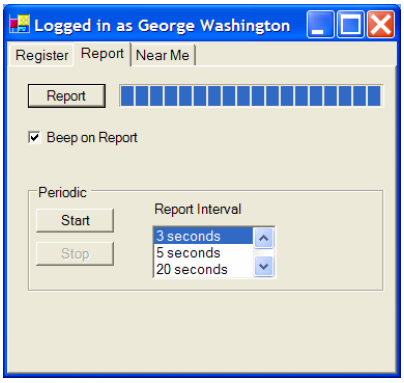
3.1 Register with Proximity Server

The user's first step in using the proximity server is to register with a chosen name, as shown in Figure 1-a. New users can type in any name, and they also chose an expiration interval in hours as well as a uniform resource locator (URL) that others can use to look up more information. The expiration interval serves as a trigger for the server to automatically delete old users. More importantly, it allows a user's name to be automatically removed from the server to help preserve privacy after he or she is no longer using the server. One scenario we envision is that a user will register with the server at the beginning of a meeting in order to find the names of other people in the same room. Since this user knows the meeting will end in one hour, he sets the expiration interval to one hour, meaning he will not need to remember to remove his name from the server after the meeting.

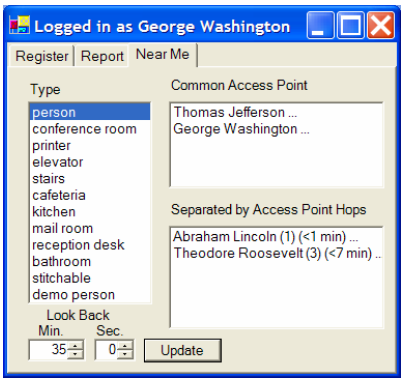
Upon registration, the client application receives a globally unique identifier (GUID) from the server. This GUID is used by the server to identify which data to associate with which user. If a user quits the client application and wants to restart later, the registration function gives him or her opportunity to register as a previous user instead of a new one. The server then responds with the GUID of the chosen previous user which is used by the client to tag future transmissions.



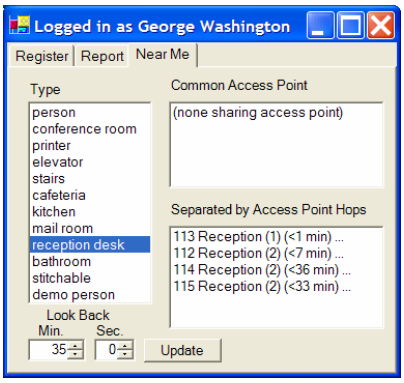
a) George Washington registers with his name and a URL. He could have also registered as one of several different places or things listed in the left column.



b) He reports his current Wi-Fi signal strengths to the server. He could optionally start a periodic sequence of reports with a chosen time interval.



c) George Washington queries for nearby people, finding Thomas Jefferson sharing an access point. Two others are some number of access point hops away, as given in the lower right list. This list gives the distance to the two other both in terms of access point hops and the minimum time it has taken anyone to walk between them.



d) He queries for receptionist desks and finds four, but none share an access point. The left list gives the various types of places that can be queried.

Figure 1: These screen shots show a typical series of actions and responses by a user of the NearMe Windows client.

A user can register as a person or as any of the possible types below:

person	elevator	kitchen	bathroom
conference room	stairs	mail room	stitchable device
printer	cafeteria	reception desk	demo person

The non-person types are intended to allow a user to tag an object or location with a Wi-Fi signature. Each registered non-person instance is given a name, just like users, but there is no expiration interval. Once tagged, human users can query the server for nearby instances of these types as well as people.

For an enterprise, an alternative, more secure registration method would be to use the username/password scheme in force for the enterprise's computer network. A wider deployment could use a publicly accessible authentication service such as Microsoft's Passport.NET. Also, it would be valuable to add the ability to limit a user's visibility to just a certain group, like his or her list of instant messenger buddies.

3.2 Reporting Wi-Fi Signatures

Once registered, a client can report access points and their measured Wi-Fi signal strengths to the server as shown in Figure 1-b. The Windows client allows the user to make a one-time report or set up a periodic series at a chosen time interval. The periodic mode is intended to be used by a moving client. A client makes generic API calls to retrieve a list of access point Media Access Control (MAC) addresses (one for each detectable access point) and the associated *received signal strength indicator*s (rssi) from its 802.11 wireless device. This list is the Wi-Fi signature. We only use APs that are in infrastructure mode, not ad hoc, as infrastructure mode APs are normally static. Rssi is normally measured in decibels referred to one milliwatt, or dBm. The usual range is approximately -100 to -20 dBm, and the APIs we use report rssi as an integer. Rssi generally decreases with distance from the access point, but it is affected by attenuation and reflection, making the relationship between location and rssi complex. MAC addresses are 6-byte identifiers that uniquely identify 802.11 access points. Our clients adhere to the general recommendation that one needs to give an 802.11 network interface card (NIC) at least three seconds to scan for access points after the scan is triggered. The clients do no filtering of detected access points, so the list can contain access points associated with any network, whether or not the client has credentials to interact with them. The clients can also detect access points with no network connection that are effectively functioning as only location beacons.

The set of MAC addresses and signal strengths is the Wi-Fi signature. The client's report consists of the client's GUID and Wi-Fi signature, which we represent as

$$\{GUID, (m_1, s_1), (m_2, s_2), \text{K}, (m_n, s_n)\} \quad (1)$$

for n detectable access points, were (m_i, s_i) are the MAC address and rssi of the i^{th} detected access point respectively. These ordered pairs are not reported in any particular order.

3.3 Querying for Nearby People and Places

The last client function is to make a query for nearby people or places as shown in Figure 1-c and Figure 1-d. The user selects a type to query for, either other people or something else from the list of types, *e.g*. printer, conference room, *etc*. The server responds with two (possibly empty) lists of nearby instances of the requested type. The first list, in short range proximity, shows those instances that have at least one detectable access point in common with the querying client, sorted roughly by distance. The second list, in long range proximity, contains instances that can be reached by "hopping" through access points with overlapping coverage, sorted by the number of hops required. Some of the instances found within hopping distance are also reported with an estimate of the amount of time it would take to travel to it. Section 5 explains how we sort the list of short range proximity. Section 4 explains how we compute hops and travel times for long range proximity.

3.4 Other Clients

A web service acts as the API for accessing the NearMe database. This makes it easy to write other clients. We have a PocketPC client that duplicates the functionality of the Windows client described above. We also have an Active Server Pages (ASP) client that runs in a conventional web browser in response to a URL that has the Wi-Fi signature encoded as simple ASCII parameters. Since the web service interface to the server is based on the simple object access protocol (SOAP), any SOAP client could access the service, including those running on Linux and MAC OS.

4 The NearMe Server

The NearMe server is a SQL database that maintains tables of active users, static resources (like printers and conference rooms), and their associated Wi-Fi signatures. It also maintains metric and topological data about the physical layout of access points derived from Wi-Fi signatures. It uses these tables to respond to client requests posed through an API in the form of a web service. The rest of this section describes the major elements of the NearMe server.

4.1 Scan Sources

Scan sources are people or places that can be associated with Wi-Fi signatures. Along with a scan source type, each scan source is represented with a GUID, a friendly name, an optional URL, an optional email address, and an expiration time for people. The NearMe server checks for expired scan sources every hour and deletes their names.

4.2 Wi-Fi Signatures

Wi-Fi signatures are lists of MAC addresses of infrastructure mode access points and their associated signal strengths generated on the client device. On the server, each Wi-Fi signature is tagged with the GUID of its scan source and a sever-generated time stamp. Wi-Fi signatures are never deleted, even if their associated scan source is deleted due to expiration. Because they are only identified with the GUID of the scan source, such orphaned signatures cannot be traced back to their originating scan source. We preserve all the Wi-Fi signatures in order to compute tables describing the layout of access points, described next.

4.3 Access Point Layout

Time-stamped Wi-Fi signatures are a valuable source of information regarding the physical layout of access points. Layout information can in turn be used to aid the computation of long range proximity. The NearMe server processes the Wi-Fi signatures in two ways.

First, the server computes the topology of the access points by examining which pairs of access points have been detected simultaneously by the same client. This indicates that the access points have physically overlapping coverage and are therefore considered adjacent. Note that adjacent access points do not have to be on the same network backbone nor even on any backbone at all. Conceptually, the NearMe server builds an adjacency matrix of access points with overlapping coverage. From this matrix, it computes an undirected graph with access points as nodes and edges between adjacent nodes. In reality, the server computes a table of pairs of access points and the minimum number of edges or hops between them, up to some maximum number of hops (currently eight). Our server is programmed to recompute this table every hour in order to keep up to date with the latest Wi-Fi signatures. In this way, the physical scope of NearMe automatically grows as more users report Wi-Fi signatures from more locations. This table is used to find people or things in long range proximity of a client, where long range indicates that the two share no detectable access points but can be connected by some number of hops between adjacent access points. The number of hops is reported to clients to give the user a rough idea of the distance to a scan source in long range proximity.

This table of adjacent access points is also used as an anti-spoofing guard. Clients can be optionally programmed with a web service call that checks to see if the access points in a Wi-Fi signature have ever before been seen together by any other client. If they have not, this raises the suspicion that the Wi-Fi signature is not valid and that it was created artificially. While this anti-spoofing check helps maintain the integrity of the database, it also prevents any growth in the list of adjacent access points, so it is only used on untrusted clients.

The second piece of layout information concerns the metric relationship between access points, and it comes from the time stamps on the Wi-Fi signatures. These are used to find the minimum transit times between pairs of access points, which can give a user an idea of how long it will take to travel to someone or something that appears

on the long range proximity list. Every hour, our server is programmed to create groups of Wi-Fi signatures that share the same GUID, meaning they came from the same scan source (*e.g.* the same person). It constructs all possible unique pairs of access points within each group. For each member of each pair, the server looks up their respective time stamps and assigns the resulting time interval to the pair. All these pairs are recombined, where all but the minimum time interval is kept for duplicate pairs. The result is a list of MAC address pairs and the minimum time any client was able to transition between them. These times are included in the list of scan sources in long range proximity, as shown in Figure 1 c-d. The times serve as an upper bound on how long it would take to travel directly to that scan source. It is an upper bound because we cannot guarantee that the minimum time observed actually came from a direct traverse between the two access points. A more sophisticated version of this analysis could cluster travel times between access points to account for the different speeds of different possible modes of transportation, like walking, biking, and driving.

Both the topological and metric tables provide valuable proximity information and are computed automatically without any extra calibration work required from either the human clients nor the system maintainer. All the data for these tables is contributed by human users, but their data is anonymized by default after expiration. We envision this type of proximity information to be used to find people and places that might typically be out of range of one access point, like a receptionist desk in a large office building, a cafeteria, a friend on campus, or a custodian. The travel time data would be useful for picking the nearest of the requested items as well as to plan how much time to allow to reach it.

The long range proximity tables are computed based on *all* past data submitted to the server. If access points in the environment are removed or added, long range proximity computations will still be valid. Moving an access point, especially to another part of the topology, would create invalid graph links. One solution we have not implemented is to expire Wi-Fi signatures older than a certain threshold.

As of this writing, our database has 1123 unique access points recorded from around our institution. On average, each access point overlaps with 16.6 other access points. The average number of access points per Wi-Fi signature is 6.1.

Our database of access points is similar in some ways to those used for Intel Research's Place Lab initiative [3] and publicly accessible "war driving" databases like NetStumbler [20] and WiGLE [21]. The main difference is that our database is not dependent on traditional war driving where access point data must include absolute locations. Instead, our database is built up in the normal course of using our clients, with the only ground truth data being the names of locations of interest, like printers and conference rooms. Thus NearMe has a lower barrier to entry, albeit at the expense of not giving absolute locations. The more traditional war driving databases could be easily adapted to work with NearMe. Indeed, one of the NearMe clients allows the database to be updated from a war driving log file. An interesting question is how NearMe could benefit from the addition of some absolute location data.

5 Range Approximation for Short Range Proximity

People and places within short range proximity of a client are defined as those that share at least one access point with the client. In computing the short range list on the server, it is useful to sort the list by distance from the client. Then a user can, for instance, pick the nearest printer or pick the N nearest people. If NearMe were a location-based system, then sorting by distance would be an easy matter of computing Euclidian distances and sorting. However, since we intentionally avoid the computation of absolute location, we must find another way.

Intuitively, the distance between two scan sources should be related to the similarity of their Wi-Fi signatures. If they see several access points in common, and if the signal strengths from those access points are similar, then it is more likely that the two are nearby each other. We designed an experiment to see how accurately we could compute the distance between clients and which features of the Wi-Fi signatures were best to use.

5.1 Similarity Features

Suppose the two Wi-Fi signatures from clients a and b are

$$\left\{\left(m_1^{(a)}, s_1^{(a)}\right), \left(m_2^{(a)}, s_2^{(a)}\right), \ldots, \left(m_{n_a}^{(a)}, s_{n_a}^{(a)}\right)\right\} \text{ and } \left\{\left(m_1^{(b)}, s_1^{(b)}\right), \left(m_2^{(b)}, s_2^{(b)}\right), \ldots, \left(m_{n_b}^{(b)}, s_{n_b}^{(b)}\right)\right\}.$$

The m's are the AP MAC addresses, and the s's are the associated signal strengths. Client a detected n_a access points and client b detected n_b. In order to define similarity features, we first form the set of access points that were detected by both clients and the associated signal strengths from each client:

$$\left\{\left(m_{\cap,1}, s_{\cap,1}^{(a)}, s_{\cap,1}^{(b)}\right), \left(m_{\cap,2}, s_{\cap,2}^{(a)}, s_{\cap,2}^{(b)}\right), \ldots, \left(m_{\cap,n_\cap}, s_{\cap,n_\cap}^{(a)}, s_{\cap,n_\cap}^{(b)}\right)\right\}$$

Here there were $n_\cap$ access points that were detected by both clients, the i^{th} of which was $m_{\cap,i}$, which clients a and b measured at signal strengths $s_{\cap,i}^{(a)}$ and $s_{\cap,i}^{(b)}$, respectively.

Our goal was to find a numerical function of the two Wi-Fi signatures that gives the physical distance separating the two clients. We first had to create numerical features from the two signatures that we thought might be useful for computing distance. The four features we experimented with are:

1. The number of access points in common between the two clients, represented by $n_\cap$. We expect that an increased $n_\cap$ is an indication of shorter range.
2. The Spearman rank-order correlation coefficient [22], denoted by ρ_s. This number represents how closely the two clients ranked their common access points by signal strength. Intuitively, a more similar ranking indicates the clients are closer together. The ranking approach was inspired by the RightSPOT system [23], which uses

ranking of FM radio station signal strengths to classify a small device into one of a discrete set of locations. The advantage of ranking is that different radio recievers, such as the Wi-Fi NICs in our clients, may well measure signal strengths in different ways. The ranking of the access points by signal strength will be the same on both clients if they both receive the same signal strengths and they both have a monotonic function relating input and measured signal strengths. While this ignores information contained in the absolute signal strengths, it is robust to inevitable variations in NICs, including differences in design, manufacturing, shielding, and antenna orientation. Mathematically ρ_s is computed by first making two sorted lists of the signal strengths seen in common by both clients. For example, these lists might be $\left(s_{\cap,1}^{(a)}, s_{\cap,2}^{(a)}, s_{\cap,3}^{(a)}\right) = (-70,-50,-80)$ and $\left(s_{\cap,1}^{(b)}, s_{\cap,2}^{(b)}, s_{\cap,3}^{(b)}\right) = (-90,-60,-70)$. In each list, we replace each signal strength with the ascending rank of that signal strength in its own list to make two rank lists, *e.g.* $\left(r_1^{(a)}, r_2^{(a)}, r_3^{(a)}\right) = (2,3,1)$ and $\left(r_1^{(b)}, r_2^{(b)}, r_3^{(b)}\right) = (1,3,2)$. The Spearman ρ_s is given by[22]:

$$\rho_s = \frac{\sum_i \left(r_i^{(a)} - \bar{r}^{(a)}\right)\left(r_i^{(b)} - \bar{r}^{(b)}\right)}{\sqrt{\sum_i \left(r_i^{(a)} - \bar{r}^{(a)}\right)^2} \sqrt{\sum_i \left(r_i^{(b)} - \bar{r}^{(b)}\right)^2}} \tag{2}$$

where $\bar{r}^{(a)}$ and $\bar{r}^{(b)}$ are the means of the ranks. In our example, $\rho_s = 0.5$. ρ_s ranges from -1 to 1, indicating poor to exact correlation between rankings, respectively.

3. Sum of squared differences of signal strengths:

$$c = \sum_i \left(r_i^{(a)} - r_i^{(b)}\right)^2 \tag{3}$$

A smaller value of c indicates more similar signal strengths and presumably shorter range. This does not account for the variability in measuring signal strengths that the ranking coefficient ρ_s is intended to ignore.

4. Number of access points unaccounted for in each list. This indicates the number of "left over" access points that are not in the list of common access points, $n_u = n_a + n_b - 2n_\cap$. More unaccounted for access points could indicate that the clients are farther apart.

5.2 Range Experiment

We gathered two sets of Wi-Fi signatures with known distances between scans. One set, used for training, was taken on one floor of a normal office building. The other set, used for testing, was taken in a cafeteria. We walked to various locations in both venues, simultaneously logging Wi-Fi signatures and the device's approximate location by clicking on a building floor plan. In order to test the effect of different Wi-Fi

NICs, we gathered data from six different ones:

Dell TrueMobile 1150 Series (built in to laptop)	ORiNOCO (PC Card)	Cisco Aironet 340 Series (PC Card)
Microsoft Wireless (USB Adapter)	Actiontec 802.1b Wireless Adapter (USB Adapter)	Linksys 802.11b Wireless (USB Adapter)

For each of the two venues, we created pairs of Wi-Fi signatures using the location stamps to determine their Euclidian separation distances in meters. We eliminated those pairs that were taken with the same Wi-Fi NIC in order to test the more realistic situation that the two Wi-Fi signatures will come from different NICs. In the office building data set, we gathered a total of 2218 Wi-Fi signatures and created 1,441,739 pairs of Wi-Fi signatures after eliminating those pairs created with the same NIC. For the cafeteria data set, we took 1038 Wi-Fi signatures and created 572,027 pairs.

Our goal was to find a function that takes some or all of the features of a pair of Wi-Fi signatures from Section 5.1 and returns an estimate for the physical distance between them. We chose polynomials as our functions, as there are no well-established physical models that relate our features and distance. For our experiment we varied the order of the polynomials, N_o, from one to four, and we varied the number of features, N_f, from one to four. For each N_f, we tested all $\binom{4}{N_f}$ ("4 choose N_f") possible combinations of features. For example, if $N_o = 2$, $N_f = 3$, and the three features were $n_\cap$, ρ_s, and c, then the polynomial would be

$$\begin{aligned} d = {} & a_{000} + a_{100} n_\cap + a_{010} \rho_s + a_{001} c + \\ & a_{110} n_\cap \rho_s + a_{101} n_\cap c + a_{011} \rho_s c + \\ & a_{200} {n_\cap}^2 + a_{020} {\rho_s}^2 + a_{002} c^2 \end{aligned} \tag{4}$$

where d is the physical distance between the locations at which the two Wi-Fi signatures were taken, and the a's are the coefficients we estimated using least squares. In computing the coefficients, we used weighted least squares to equalize the influence of each possible pair of NICs, because each NIC was not represented exactly equally in the experimental data.

We used the office building data as training data to compute polynomial coefficients. Because of the large number of data points, we performed the actual least squares fitting on 10 subsets each consisting of a random 10% of the data, and we kept the coefficients that gave the minimum rms distance error from each subset. The results are shown in Table 1. For the training data, the rms error was in the vicinity of 7 meters, with a minimum of 6.43 meters for the 3rd degree polynomial using all four features. We also evaluated how well the computed polynomials ranked the distances using the Spearman rank correlation coefficient between the actual and computed ranked distances. (Note that we use Spearman twice: once as a way to measure the

Number of Features	Feature(s)	Polynomial Degree	RMS Err (m)		Spearman ρ	
			Train	Test	Train	Test
1	APs In Common	1	7.13	14.23	-0.36	0.30
		2	7.25	14.22	-0.36	0.30
		3	7.13	14.24	0.32	0.30
	Spearman ρ	1	7.26	14.85	0.19	0.19
		2	7.22	14.67	0.17	0.22
		3	7.20	14.63	0.19	0.19
	RSSI Difference	1	7.58	15.09	-0.27	0.26
		2	7.63	15.08	-0.26	0.26
		3	7.44	15.04	0.33	0.29
	Unaccounted for APs	1	7.23	15.23	0.31	0.30
		2	7.16	15.24	0.31	0.30
		3	7.09	15.13	0.31	0.30
2	APs In Common Spearman ρ	1	6.83	14.04	0.38	0.39
		2	6.75	14.19	0.41	0.34
		3	6.74	14.26	0.41	0.32
	APs In Common RSSI Difference	1	7.10	14.31	0.39	0.39
		2	7.12	14.38	0.22	0.39
		3	6.96	14.16	0.40	0.39
	APs In Common Unaccounted for APs	1	6.87	14.57	0.39	0.35
		2	6.83	14.78	0.40	0.35
		3	6.83	14.80	0.40	0.34
	Spearman ρ RSSI Difference	1	7.24	14.80	0.15	0.26
		2	7.24	14.68	0.12	0.09
		3	7.07	14.68	0.29	0.37
	Spearman ρ Unaccounted for APs	1	7.00	15.08	0.33	0.31
		2	6.91	14.91	0.36	0.33
		3	6.83	14.99	0.36	0.33
	RSSI Difference Unaccounted for APs	1	7.16	15.26	0.28	0.23
		2	7.10	15.22	0.28	0.22
		3	6.91	15.07	0.41	0.34

Number of Features	Feature(s)	Polynomial Degree	RMS Err (m)		Spearman ρ	
			Train	Test	Train	Test
3	APs In Common Spearman ρ RSSI Difference	1	6.81	13.97	0.35	0.43
		2	6.75	14.13	0.37	0.30
		3	6.61	14.10	0.44	0.38
	APs In Common Spearman ρ Unaccounted for Aps	1	6.69	14.20	0.43	0.41
		2	6.58	14.42	0.45	0.38
		3	6.52	14.47	0.46	0.35
	APs In Common RSSI Difference Unaccounted for APs	1	6.88	14.39	0.38	0.33
		2	6.88	14.55	0.38	0.30
		3	6.70	14.51	0.44	0.36
	Spearman ρ RSSI Difference Unaccounted for Aps	1	7.01	15.00	0.33	0.30
		2	6.91	14.89	0.36	0.29
		3	6.68	14.83	0.42	0.35
4	APs In Common Spearman ρ RSSI Difference Unaccounted for APs	1	6.71	14.30	0.42	0.40
		2	6.64	14.60	0.44	0.35
		3	6.43	14.49	0.49	0.36

Table 1: Results of training and testing polynomials to estimate distance from Wi-Fi signatures. The "Train" column under the "RMS Err (m)" column shows the rms error in meters after the least squares fit to the office building data. The "Train" column under "Spearman ρ" shows how well the computed polynomial ranked the computed distances compared to the actual distances. The two "Test" columns show how well the office building polynomial coefficients worked on the cafeteria data. In general, increasing the number of features and the degree of the polynomial did not significantly improve accuracy.

rank similarity of signal strengths and once as a way to assess how well our various polynomials rank physical distance compared to ground truth.) This is useful since some applications may want to present ranked lists of nearby people rather than their absolute distances. The maximum Spearman correlation for the training set was 0.49, also for the 3rd degree polynomial using all four features.

We used the polynomial coefficients from the office building training set to see how well they worked for the cafeteria data set. This gives us an idea of whether or not we could put forth a broad recommendation for which features and functions to use for any general situation. This will require more testing in the future, but the cafeteria data shows reasonable performance with a minimum rms error of 13.97 meters and a maximum Spearman correlation of 0.43, both using a 1st degree polynomial on $n_{\cap}$, ρ_s, and c. The number of unaccounted for access points, n_u, was the worst performing single feature in terms of rms error on the test set. Intuitively, the most attractive features are $n_{\cap}$ (the number of access points in common) and ρ_s (the Spearman correlation of the signal strengths), because they are robust to measurement differences between NICs. The test data indicates that the best performing polynomial for these two features was a 1st degree polynomial, giving an rms error of 14.04 meters and a Spearman correlation of 0.39, both very close to the best performance over all the test cases. The actual polynomial was

$$d = -2.53 n_{\cap} - 2.90 \rho_s + 22.31 \quad (5)$$

As expected, this equation indicates that the estimated distance in meters (d) decreases when more access points are seen in common ($n_{\cap}$) and when their relative rankings are more similar (ρ_s). One interesting aspect of this equation is that $\partial d / \partial n_{\cap} \approx \partial d / \partial \rho_s$, meaning that $n_{\cap}$ and ρ_s have approximately the same level of influence on the estimated distance. Given this similarity in influence, if the goal is to sort Wi-Fi signature pairs by distance, a reasonable heuristic is to simply sort by the sum $n_{\cap} + \rho_s$. This is what we do on the server to sort lists of instances in short range proximity.

Although this equation worked reasonably well for our two data sets, the actual coefficients are likely not broadly applicable to other locations where there could be differences in building materials, architecture, access point density, and access point transmission strength. One example of its possible inapplicability is in an area densely populated with access points. In such a case, $n_{\cap}$ could be large enough that the computed distance is negative. However, this analysis does indicate which features of the Wi-Fi signatures are important, and it leads to the $n_{\cap} + \rho_s$ heuristic for sorting by distance. No calibration is necessary to apply this heuristic to a new environment, in contrast to Wi-Fi location systems that normally require manually constructed or simulated radio maps. These calibrated systems do provide more accuracy, however, with median absolute location errors of 2.37 meters for RADAR[2], 1.53 meters for LOCADIO[24], and 1 meter for the system of Ladd *et al.*[25]. For proximity, this level of accuracy is not always necessary.

Short range proximity computations are robust to the addition and deletion of access points, because the distance computation is based on only the list of access points that two Wi-Fi signatures have in common. A moved access point could cause large errors. However, for finding nearby people who are updating their Wi-Fi signatures frequently, as our basic client allows (Figure 1b), even moved access points are easily tolerated.

6 Applications

The functionality of the NearMe server is exposed as a web service, making it easy to create new clients. This section describes three potentially useful clients.

6.1 Sample Client with URLs

The sample client in Figure 1 allows people and places to be registered with a URL. For example, people might register with their home pages. For some places, like reception desks, we registered a URL giving a map to help visitors find their way. Instances with a registered URL show up on the proximity lists with a "…" behind their name. The user can click on these names to bring up a web browser showing their

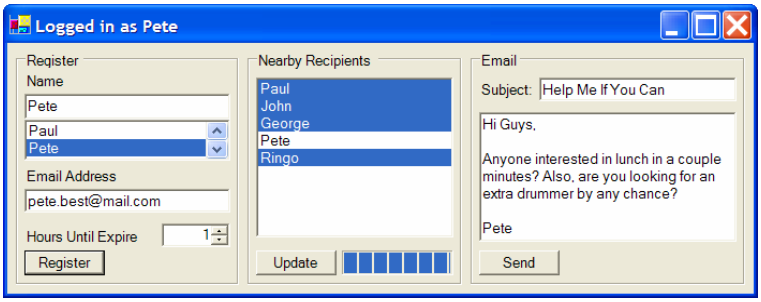


Figure 2: This program allows a user to send email to other users in proximity.

URLs. Each registered person and place is essentially tagged with a Wi-Fi signature that serves for filtering based on location. The changing lists of proximal people and places, along with their associated URLs, create a dynamic lookup service of what is available nearby.

6.2 Localized Email

The screen shot in Figure 2 shows our localized email program. It allows a user to register with NearMe with a name and email address. After updating the database with his or her Wi-Fi signature, a list of nearby registered users appears. The user can select names from this list and send an email to them. This would be useful for nearly immediate requests like going out to lunch or asking for face-to-face help with a problem. Because we sort the list of potential recipients by physical distance, picking the top N in the list is equivalent to picking the N nearest people, up to NearMe's inherent distance approximation errors. Since NearMe's range resolution is in the tens of meters, its errors are likely tolerable for this application. In the future, proximity could be one of a number of filters for email recipients, optionally used in addition to filters on recipient type (*e.g.* friend, colleague, supervisor) and interest area.

6.3 Detecting Synchronous User Operations

Another client we have implemented uses NearMe to aid in detecting synchronous user operations between mobile devices for co-located collaboration. Stitching [17], synchronous gestures [14], and SyncTap [16] are all examples such techniques. Stitching, for example, must share the screen coordinates, direction, and timing of pen strokes with other nearby devices to establish when a pen stroke spans the displays of two devices. This makes it easy for users to drag a file between two separate pen-operated wireless devices, for example, as shown in Figure 3.

A key problem in this class of systems is to determine which devices to consider as candidates for potential synchronous user operations [14]. SyncTap [16] proposes using multicast to share user activity and timing with other devices, but this may needlessly send information to a large number of irrelevant devices that are too far apart to ever be intentionally associated. Restricting communications to devices that are truly

nearby reduces the potential for false-positive recognition of synchronous user operations (due to pure chance synchronization of incidental operations on a large number of devices) and also may help to reduce power consumption requirements (by reducing wasted computation and transmission of messages seeking to establish synchronization with non-proximal devices).

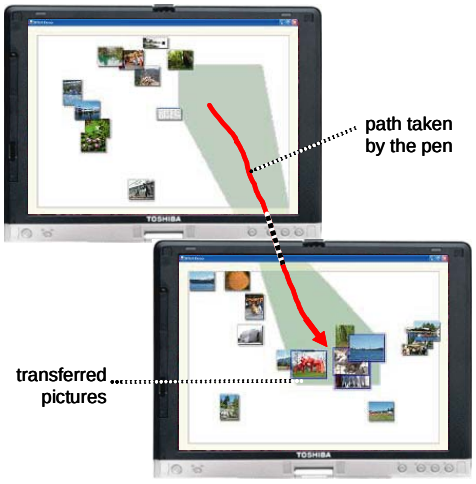


Figure 3: NearMe is used to find which devices are in proximity so they can be associated with a pen gesture spanning both screens.

NearMe solves these problems by providing a list of nearby devices for every device that seeks associations with other devices. For our Stitching technique, we refer to these as "stitchable devices." Our Stitching system software only looks for correlating pen strokes within sets of devices that NearMe identifies as being within short range proximity of one another. Stitchable devices update their signal strengths with NearMe every 20 seconds so that the set of stitchable devices at any one time is dynamic and discoverable by any new client wishing to make itself eligible for stitching. While this application considers associations for any device within short range proximity, it could be modified to consider only those devices within some physical range based on our distance estimation. But even as implemented, NearMe reduces the list of potentially associable devices from the whole world to just those within the range of one access point.

7 Conclusions

NearMe's main feature is that it gives lists of nearby people and places without computing their absolute locations. This makes it easier to deploy than traditional location-based systems. Even though it is unaware of absolute locations, NearMe can still give absolute and relative distance estimates for short range proximity, and it can give travel time estimates for long range proximity. The database grows as more people use the client, which in turn increases the richness and range of people and places that can be found in proximity. The database helps protect the privacy of users by anonymizing their data after a user specified time period, and it can protect itself against falsified access point signatures by verifying them against what it has already seen.

As this work proceeds, we would like to test the feasibility of using NearMe in a peer-to-peer fashion rather than depending on a central database. For short range

proximity, this would be a simple matter of having peers exchange Wi-Fi signatures and then having the clients evaluate our function for estimating separation distance. Another way to expand the scope of NearMe would be to incorporate other types of radio as location signatures, such as Bluetooth, cell towers, and commercial broadcasts of radio and TV.

References

1. Hightower, J. and G. Borriello, *Location Systems for Ubiquitous Computing*. Computer, 2001. **34**(8): p. 57-66.
2. Bahl, P. and V.N. Padmanabhan. *RADAR: An In-Building RF-Based User Location and Tracking System*. in *INFOCOM 2000*. 2000.
3. Schilit, B.N., et al. *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)*. 2003. San Diego, California, USA.
4. Schilit, B.N., N. Adams, and R. Want. *Context-Aware Computing Applications*. in *IEEE Workshop on Mobile Computing Systems and Applications*. 1994.
5. Hightower, J., G. Borriello, and D. Fox, *The Location Stack*. 2003, Intel Research Seattle. p. 13.
6. http://www.ntag.com/.
7. http://www.spotme.ch/.
8. Cox, D., V. Kindratenko, and D. Pointer. *IntelliBadge™: Towards Providing Location-Aware Value-Added Services at Academic Conferences*. in *UbiComp 2003: Ubiquitous Computing*. 2003. Seattle, WA, USA.
9. Dey, A.K., et al. *The Conference Assistant: Combining Context-Awareness with Wearable Computing*. in *3rd International Symposium on Wearable Computers*. 1999. San Francisco, California, USA.
10. Dahlberg, P., F. Ljungberg, and J. Sanneblad. *Supporting Opportunistic Communication in Mobile Settings*. in *CHI 2000 Extended Abstracts on Human Factors in Computing Systems*. 2000. The Hague, The Netherlands: ACM Press.
11. Sumi, Y. and K. Mase. *Digital Assistant for Supporting Conference Participants: An Attempt to Combine Mobile, Ubiquitous and Web Computing*. in *Ubicomp 2001: Ubiquitous Computing*. 2001. Atlanta, Georgia, USA: Springer.
12. Woodings, R., et al. *Rapid Heterogeneous Ad Hoc Connection Establishment: Accelerating Bluetooth Inquiry Using IrDA*. in *Third Annual IEEE Wireless Communications and Networking Conference (WCNC '02)*. 2002. Orlando, Florida, USA.
13. Holmquist, L.E., et al. *Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts*. in *Ubicomp 2001: Ubiquitous Computing*. 2001. Atlanta, Georgia, USA: Springer.
14. Hinckley, K. *Synchronous Gestures for Multiple Users and Computers*. in *UIST'03 Symposium on User Interface Software & Technology*. 2003.
15. Lester, J., B. Hannaford, and G. Borriello. *"Are You With Me?" – Using Accelerometers to Determine if Two Devices are Carried by the Same Person*. in *Pervasive 2004*. 2004. Linz, Austria.
16. Rekimoto, J., Y. Ayatsuka, and M. Kohno. *SyncTap: An Interaction Technique for Mobile Networking*. in *Mobile HCI*. 2003.
17. Hinckley, K., et al. *Stitching: Pen Gestures that Span Multiple Displays*. in *ACM Advanced Visual Interfaces (AVI 2004)*. 2004.
18. http://www.trepia.com/.
19. http://www.apple.com/macosx/features/rendezvous/.
20. http://www.netstumbler.com/.
21. http://www.wigle.net/.
22. Press, W.H., et al., *Numerical Recipes in C*. 1992, Cambridge: Cambridge University Press.
23. Krumm, J., G. Cermak, and E. Horvitz. *RightSPOT: A Novel Sense of Location for a Smart Personal Object*. in *UbiComp 2003: Ubiquitous Computing*. 2003. Seattle, WA: Springer.
24. Krumm, J. and E. Horvitz. *LOCADIO: Inferring Motion and Location from Wi-Fi Signal Strengths*. in *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous 2004)*. 2004. Boston, MA.
25. Ladd, A.M., et al. *Robotics-Based Location Sensing using Wireless Ethernet*. in *International Conference on Mobile Computing and Networking*. 2002. Atlanta, GA: ACM Press.