World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:1, No:9, 2007

WiPoD Wireless Positioning System based on 802.11 WLAN Infrastructure

Halûk Gümüşkaya, and Hüseyin Hakkoymaz

***Abstract*—**This paper describes WiPoD (Wireless Position Detector) which is a pure software based location determination and tracking (positioning) system. It uses empirical signal strength measurements from different wireless access points for mobile user positioning. It is designed to determine the location of users having 802.11 enabled mobile devices in an 802.11 WLAN infrastructure and track them in real time. WiPoD is the first main module in our LBS (Location Based Services) framework. We tested K-Nearest Neighbor and Triangulation algorithms to estimate the position of a mobile user. We also give the analysis results of these algorithms for real time operations. In this paper, we propose a supportable, i.e. understandable, maintainable, scalable and portable wireless positioning system architecture for an LBS framework. The WiPoD software has a multithreaded structure and was designed and implemented with paying attention to supportability features and real-time constraints and using object oriented design principles. We also describe the real-time software design issues of a wireless positioning system which will be part of an LBS framework.

***Keywords*—**Indoor location determination and tracking, positioning in Wireless LAN.

I. Introduction

LOCATION based services (LBS) is a relatively recent branch of mobile networking, which started to expand rapidly after mobile networks had been enabled to determine the locations of mobile devices. LBS help users interact better with their environment. LBS applications provide location (friends, nearest printer, nearest restaurant), navigation (in-building, metro area), information, targeted advertising (sales, election canvassing), notification (buddy alert, weather alert), and other services, where the awareness of user location is being critical [1]–[3]. LBS applications may be viewed as being built over positioning systems extending them to providing task specific location-relative information. An LBS system may be characterized by underlining infrastructure, positioning method, and its application speciation. User positioning is the first prerequisite to an LBS framework.

Wireless positioning using IEEE 802.11 WLAN (Wireless Local Area Network) technologies has been a hot topic in the recent years. RADAR from Microsoft [4] and [5], [6] and Ekahau Positioning Engine [7] are examples of such wireless positioning systems. In this paper, we present the implementation and performance evaluation of WiPoD, (Wireless Position Detector), which is the main building block of our LBS framework. We investigated the possible implementations of WLAN positioning using different algorithms and if the implementation could be achieved at a level of acceptable accuracy to be used in real-life cases. We also worked on the real-time software design issues of a wireless positioning system which will be part of an LBS framework. We propose a supportable, i.e. understandable, maintainable, scalable and portable wireless positioning software architecture for an LBS framework.

Manuscript received October 14, 2005.

Halûk Gümüşkaya is with the Department of Computer Engineering at Fatih University, Istanbul, Turkey. (phone: +90-212-8890810; fax: +90-212-8890906 ; e-mail: haluk@fatih.edu.tr).

Hüseyin Hakkoymaz is with the Department of Computer Science and Engineering at the University of California, Riverside, USA (e-mail: huseyin@cs.ucr.edu).

II. WiPoD System and Positioning Methodology

A. WiPoD System

WiPoD is a wireless positioning system developed in the Computer Engineering Department at Fatih University [8]. The system locates and tracks the user having an IEEE 802.11 supported device across the coverage area of a WLAN.

Our first experimental testbed was located on the first floor of the cafeteria at Fatih University. The floor has a dimension of 17 m by 18 m, an area of 306 square meters. We placed three access points, AP1, AP2 and AP3, at the three corners of the cafeteria. The access points provide overlapping coverage in portions of the floor, and together cover the entire floor. Our mobile host, carried by the user being tracked, was a laptop computer running Microsoft XP. The mobile host was equipped with a Cisco Aironet 802.11a/b/g PCMCIA card.

WiPoD can use any existing WLAN infrastructure and therefore, does not require any specialized hardware. WiPoD is a software-only positioning system built over an off-the-shelf WLAN. It operates by recording and processing signal strength information from multiple access points. User positioning process in WiPoD consists of two phases: Data collection phase and real time phase.

B. Data Collection Phase

In this phase, the signal strength data together with the physical coordinates of each location are collected and a signal database which we call a radio map that will be used in the real time phase is constructed. During this phase, the user walks in the testbed area and takes samples with the help of WiPoD Spotter as shown in Fig. 1.

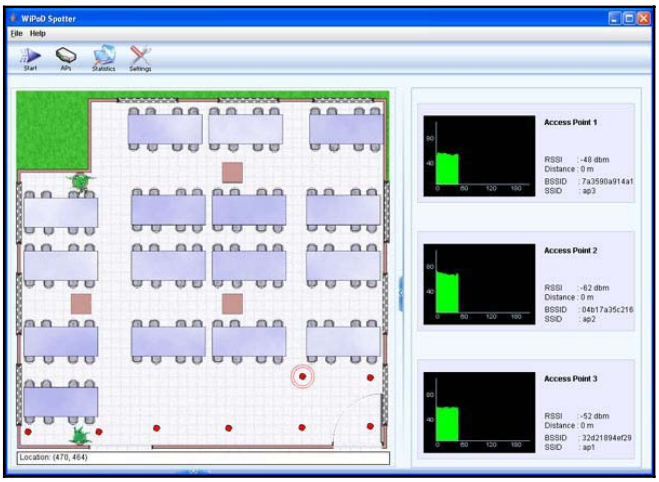


Fig. 1 WiPoD Spotter, wireless data collection and utility program

We developed WiPoD Spotter as a wireless data collector and utility program. It is especially useful for wireless programmers and researchers for analysis. It collects information about the wireless low level environmental data. First, it detects the access points and their SSID (Service Set Identifier), BSSID (Basic Service Set Identifier), RSS (Receive Signal Strength) across the coverage are of the testbed. It continuously draws RSS data for each access point as real-time signal in a window area on the right panel of WiPoD Spotter as shown in Fig. 1.

This program also constructs a radio map of the testbed area. The user loads the floor layout of the target testbed from the menu. Our first testbed cafeteria floor layout is also shown in Fig. 1 in the left panel of WiPoD Spotter. The dots denote locations where empirical signal strength information is being collected. The mobile user walks and stops in several different locations in the testbed. When stopped, he clicks the current location on the map. After the map is clicked, the WiPoD Spotter records the physical coordinates of the current location together with the signal strength of beacon packets from each of the APs within range. This operation is repeated for other locations in the testbed. This all process is repeated a couple of times to obtain a stable radio map in different days. The mean of the collected signal strength data is calculated for each AP and recorded to the database along with the coordinates.

In the creation of the radio map, the tuples will be of the form: $(x, y, \{S_1, S_2, S_3\})$, where x, y refer to the x and y coordinates of a physical point on the map, and $\{S_1, S_2, S_3\}$ refers to a reading of signal strength from each of the access points within the wireless network. We recorded 35 signal measurement points in the testbed. For each measurement point, we took 15 different measurements from each access point; we then calculated the average, minimum, maximum, signal strengths and also standard deviation for each of them. According to these data, we produced our radio map to be used in K-Nearest Neighbor algorithm and a distance estimation model for the triangulation algorithm.

C. Real Time Phase

In this phase, the location of a mobile user is determined and the path of user is tracked using WiPoD Tracker which is our second program for real time user tracking. The mobile user only needs the radio map and the layout map of the building which can be download the first time when entering the building from our LBS server. WiPoD Tracker has a similar graphical user interface given in Fig 1 for WiPoD Spotter, but it does not have the wireless utility part shown at the right panel.

III. Algorithms, Experiments and Analysis

We tested K-Nearest Neighbor (KNN) and Triangulation (TN) algorithms in our system to estimate the position of a mobile user. The algorithms are based on the received signal strength information from access points. The general philosophy in these approaches is to establish a one-to-one correspondence between a given position and the received signal strength from at least 3 transmitters with known locations. Triangulation has been widely used by various known location systems including GPS. KNN has been first used in RADAR [4], [5].

In KNN algorithm, the mobile device measures the signal strength of each of the base stations within range; and then searches through the radio map database to determine the signal strength tuple that best matches the signal strengths, it has measured. The system estimates the location associated with the best-matching signal strength tuple (i.e. nearest neighbor) to be the location of the mobile. In our analysis, we use the following Euclidean distance measure:

$$sd = \sqrt{\sum_{i=0}^{i=k} (s_i - s_i')^2} \quad (1)$$

Using this measure, we investigate the signal space to find the position in the physical space. In the formula, k is the number of access points in the physical space. S_i is the signals match in the database for AP_i and Si' is the measured signal strength in the real-time operation for AP_i. This technique basically calculates the Euclidean distance (sd) in signal space and then picks the signal tuple that minimizes this distance in the signal space and declares the corresponding physical coordinate as its estimate of the mobile's location. Alternative strategies such as averaging the k nearest neighbors have also been considered.

Triangulation algorithm gives the position of the mobile node by estimating its distance from three access points, transforming each RSS into a range through some attenuation model in function of distance. This algorithm works with formulas that are based on the geometric properties of circles and triangles. After the estimation of distances to each access point, for each access point, algorithm draws a circle whose radius is the estimated distance and center is the physical coordinates of the access point. Then it searches for these drawn circles' intersection area which is the determined location of the mobile user.

A. Evaluation of the Algorithms

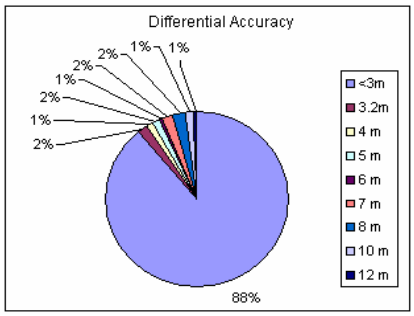
For evaluation of the algorithms, we selected 300 samples from the collected data as test data. In order to calculate the accuracy probability, we wrote a program which calculates the difference between the original location and the found location. We can feed the program with an error rate or accuracy performance as parameter and get the error rate or accuracy percentage according to this input. These two location determination algorithms are compared in Table I.

TABLE I
COMPARISONS OF ALGORITHMS

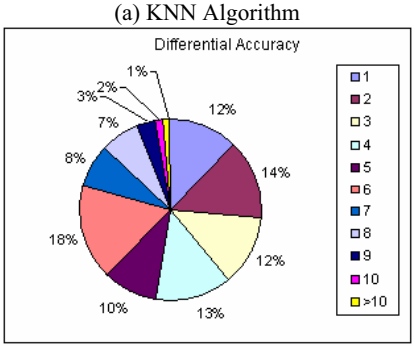
	Accuracy at the same error rate of 3.6 m	Minimum error rate at the same accuracy percentage of % 88
KNN	% 88.25	3.2 m
TNN	% 43.17	7.1 m

We tried to find the accuracy probabilities for each error rate up to 12 meters. These error rates were handled in two categories: Differential accuracy and cumulative accuracy.

In differential accuracy, the accuracy of individual error rates is calculated. For example, when we calculate the differential accuracy of 5 meters, we find the accuracy for locations only with 5 meter error rate excluding other distances. These accuracy ratios are given in Fig. 2.



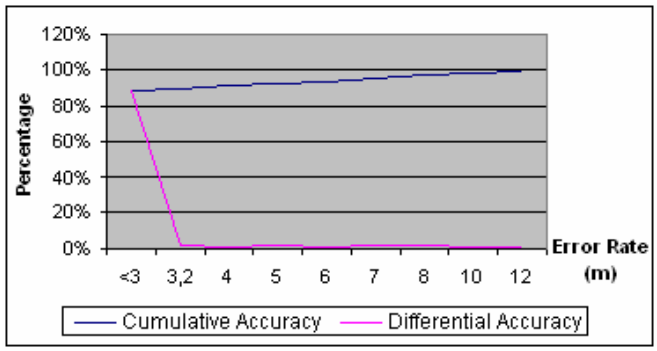
(a) KNN Algorithm



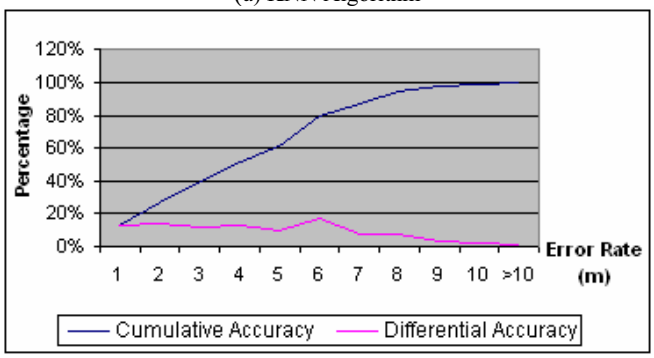
(b) TN Algorithm

Fig. 2 Differential accuracies of KNN and TN algorithms

On the other hand, cumulative accuracy includes all locations found with the error rate up to *d* meters where *d* is the error distance to the original location. For example, let *d* be 5 meters. When we ask our program for this error rate, it will accept all found locations as true if the error distance is smaller than 5 meters. When we divide these true locations with the whole 300 samples, we found the accuracy probability for each algorithm as shown in Fig 3.



(a) KNN Algorithm



(b) TN Algorithm

Fig. 3 Comparison of differential and cumulative accuracies

B. Effect of Number of Access Points

As signal strengths of radio signals emitted from different APs are the key concept in the location determination systems, the number of access points has an effect on the system performance. In order to investigate this effect, we trained the KNN algorithm with different number of access points as shown in Table II and performed four experiments up to three access points. In three of them we used the traditional algorithm and in the fourth one we added a filter function in the algorithm to train the KNN in a better way. In the first three experiments we used the radio map formed with the data including the unstable signal strengths. In the last algorithm, we eliminated these data from the database and performed the operation through these data. As the results on the table are indicating, as more access points were added into the map, more accuracy is achieved in the performance of the system.

TABLE II
EFFECT OF NUMBER OF ACCESS POINTS

	Number of Access Points			
	1 AP	2 APs	3 APs	3 APs (filtered)
Accuracy (<3m)	% 16.8	% 44.8	% 69.3	% 88.5

C. Effect of Obstructions and Reflections

Severe multipath fading and shadowing present in the indoor environment obviously have great influences on the system accuracy. The model which handles these problems

more seriously has more accuracy rate than the other one's and as shown in the diagrams and tables given above. TNN algorithm gives less accurate results compared to KNN.

During the experiments we have observed that sometimes the wireless adapter was receiving two signals from the same access point. As we investigated the situation, one signal was received from the access point directly while other one coming to the hardware after hitting the wall due to reflection. In this situation, there occurs confusion about which source will be used in calculation. To solve this problem, we compare the current signal strengths with the previous one and select the one with more similarity.

IV. Software Architecture of WiPoD

The WiPoD software architecture is shown in Fig 4 and based on a similar PCMEF layered architecture [9]. A layered architecture is a system containing multiple, strongly separated layers, with minimal dependencies and interactions between the layers. Such a system has good separation of concerns, meaning that we can deal with different areas of the application code in isolation, with minimal or no side effects in different layers. By separating the system's different pieces and following the design principles presented in section 5, we make the software understandable, maintainable, scalable, and portable, so that we can easily change and enhance it as requirements change, and port the WiPoD system to different mobile devices.

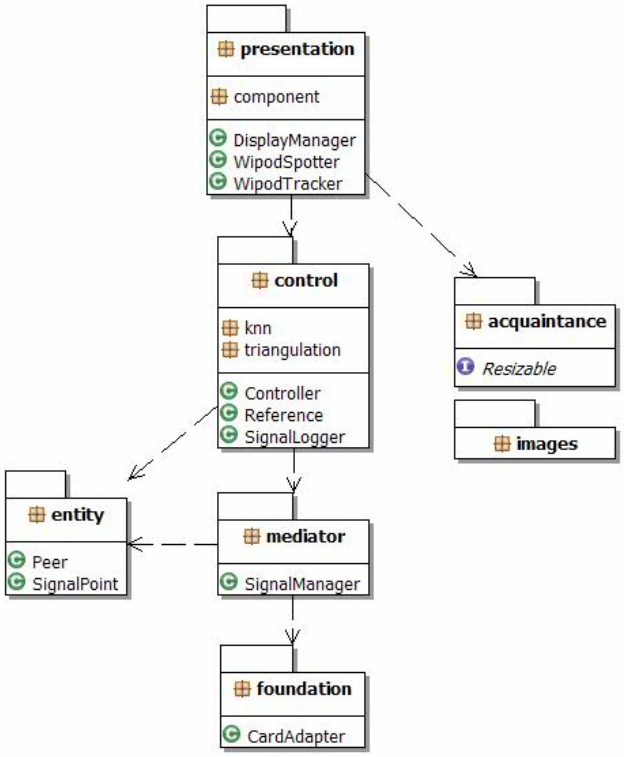


Fig. 4 WiPoD software architecture, subsystems and dependencies

The layers we are concerned with here include Presentation, Control, Mediator, Entity, and Foundation. With reference to the MVC (Model View Controller) framework, Presentation corresponds to MVC View, Control to Controller, and Entity to Model. Mediator and Foundation do not have MVC counterparts.

As shown in Fig 4, each layer communicates with only lower and upper layer. The highest layer–which is Presentation–, is the package that includes the user interface classes and it assists in the human-computer interactions. Our main classes, WiPoDSpotter and WiPoDTracker, are in this subsystem as wireless data collector and user tracker units.

Control layer is responsible for functionalities such as determining the location, creating the radio map, retrieving the data from mediator and passing this data to Presentation layer. Two location determination algorithms which are KNN and TN were implemented in this layer. In order to evaluate the accuracy of these algorithms at various physical spaces, we developed specific programs to analyze them. The evaluation results can be presented in the Presentation layer via a graphical user interface.

Mediator is the key point in the system as it mediates between hardware layer classes and high-level application layer classes. While Entity manages the wireless data objects currently in the memory, Mediator ensures that Control subsystem gets access to these data objects.

At the bottom of system hierarchy, Foundation provides a communication link between hardware, a WLAN interface card, and the Mediator. Data is retrieved from and sent to the hardware via classes that know how to talk to the wireless adapter. As other subsystems are implemented only in Java, this package includes parts which are written in both C and Java. In order to communicate with the network adapter, we used a Windows specification known as NDIS (Network Driver Interface Specification) [10] via a device driver and Protocol Manager to "bind" to the adapter and issue commands or make queries to the network adapter. For the device driver we used an API (Application Programming Interface) known as RawEther. RawEther is a framework for development of Windows products that directly access NDIS network interface drivers from Win32 applications [11]. We used the JNI (Java Native Interface) native programming interface to access the native DLL code produced by RawEther. By writing programs using the JNI, we ensure that our code is completely portable across all platforms [12], [13].

V. Design Issues for Supportable WiPoD System

The quality metrics for our LBS framework are understandability, maintainability, scalability, and portability. We call these properties the software system's supportability features. We use these quality metrics as the basic quality criteria for our system architecture.

An object oriented software system has a set of intercommunicating objects. The allowed object communication paths, defined either statically (compile-time) or dynamically (run-time), determine the possible set of object dependencies. A necessary condition to understand a system behavior is to identify and measure all object dependencies.

Dependencies can be on classes, messages, events, and inheritance. The idea is to uncover all object dependencies in a system and make them explicit. The associations are established on all directly collaborating classes in compile-time data structures. The dynamic links formed at run-time and uncontrolled polymorphic behaviors are very difficult to control and create maintenance hassle. WiPoD legitimizes run-time object communication using compile-time data structures and forbids muddy programming solutions utilizing just run-time programming structures.

Circular dependencies between layers, between packages and between classes within packages must be broken. Cycles should be resolved by creating a new package specifically to eliminate the cycle, and by forcing one of the communication paths in the cycle to communicate via interface [14].

The main dependency structure in WiPoD is top-down. Objects in higher layers depend on objects in lower layers. Consequently, lower layers are more stable than higher layers. Upward dependencies are realized through loose coupling facilitated by interfaces, event processing, acquaintance package and similar techniques. Dependencies are only permitted between neighboring layers.

The upward notification principle in WiPoD promotes low coupling in bottom-up communication between layers. This can be achieved by using asynchronous communication based on event processing. Objects in higher layers act as subscribers (observers) to state changes in lower layers. When an object (publisher) in a lower layer changes its state, it sends notifications to its subscribers. In response, subscribers can communicate with the publisher (now in the downward direction) so that their states are synchronized with the state of the publisher.

VI. Conclusion

In this paper, we have presented WiPoD supportable wireless positioning system for locating and tracking users inside a building. Our experimental results are quite encouraging. With high probability, using KNN algorithm, WiPoD is able to estimate a user's location to within a few meters of his/her actual location. This suggests that a large class of location-aware services can be built over this system.

Our future work will first focus on porting our portable software to an IPAQ Pocket PC and start to develop some LBS applications in our new wireless lab at Fatih University. We started to develop a mobile middleware and a distributed system architecture for different LBS services based on IEEE 802.11 and Bluetooth wireless infrastructures. Real-time multimedia steaming over wireless ad hoc networks will be one of the first services of our LBS framework.

Acknowledgment

The authors would like to thank Abdullah Aslan for his help in writing the C program to read wireless data from a WLAN card on a Windows XP computer and collecting the extensive RSS data used in this work, and İlker Başaran for his valuable insights.

References

[1] M. Hazas, J. Scott and J. Krumm, "Location-Aware Computing Comes of Age," *IEEE Computer*, vol. 37, no. 2, February 2004, pp. 95-97.
[2] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *IEEE Computer*, pp. 57-66, 2001.
[3] J. A. Tauber, "Location Systems for Pervasive Computing," Area Exam Report, Massachusetts Institute of Technology, August 2002.
[4] P. Bahl, V. N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracing System," *Proceedings of IEEE Infocom 2000*, Tel Aviv, Israel, March 2000.
[5] P. Bahl, V. N. Padmanabhan, "Enhancements to the RADAR User Location and Tracking System," *Microsoft Research Technical Report: MSR-TR-00-12*, February 2000.
[6] Z. Xiang, S. Song, J. Chen, H. Wang, J. Huang, X. Gao, "A Wireless LAN Based Indoor Positioning Technology", *IBM J. Res. & Dev.*, Vol. 48 No. 5/6 September/November 2004.
[7] Ekahau, Inc., Ekahau Positioning Engine, http://www.ekahau.com .
[8] H. Hakkoymaz, A. Aslan, *Wireless Location Determination System Using 802.11: Wireless Position Detector (WiPoD)*, Senior Design Project, Fatih University, İstanbul, Turkey, June 2005.
[9] L. A. Maciaszek, B. L. Liong, "Designing Measurably-Supportable Systems," *Advanced Information Technologies for Management*, Research Papers No 986, ed. by E. Niedzielska, H. Dudycz, M. Dyczkowski, pp.120-149, 2003.
[10] NDIS Developer's Reference web site: http://www.ndis.com/.
[11] RawEther for Windows web site: http://www.rawether.net/.
[12] JNI tutorial web site: http://java.sun.com/docs/books/tutorial/native1.1/.
[13] S. Liang, *The Java Native Interface: Programmer's Guide and Specification*, Addison Wesley, 1999.
[14] L.A. Maciaszek, B.L Liong, *Practical Software Engineering*, Harlow England, Addison-Wesley, 2005.