# A BLUETOOTH-BASED INDOOR LBS SYSTEM

F.Cheraghi , M.A.Rajabi

Dept. of Surveying and Geomatics Eng., University of Tehran, Tehran, P.O.Box: 14665-331, Iran
Tel: +98 21 8833 4341, Fax: +98 21 8800 8837, homepage: www.ut.ac.ir
fcheraghi@ut.ac.ir, marajabi@ut.ac.ir

**Commission VI, WG VI/4**

**KEY WORDS:** Bluetooth, WebGIS, Server-Client, Indoor Positioning, Navigation, Ubiquitous GIS, Location Based Service (LBS), Shortest Path

**ABSTRACT:**

The Location Based Service (LBS) market has been growing tremendously over the last few years. Mobility and positioning (especially indoor) have been the key features behind the success of LBS. Bluetooth is a new and promising low power, short range, wireless radio system that can be used to create a wide range of LBS services. This paper proposes a Bluetooth-based indoor LBS system using Off-The-Shelf low-cost stationary Bluetooth beacons. An application composed of three parts: Guest user, who has a pocket PC, stationary Bluetooth beacons with their unique ID (access points), and a server which is connected to the network (e.g. via Internet/intranet) is built. The system can be implemented at a low-cost which is enough for fast prototyping of applications where room-level ( ~5m) location detection is sufficient. Guest user not only can navigate through suggested list of places where they can go, but also can receive other appropriate services. This paper discusses the architecture of the developed system and its possible applications.

## 1.INTRODUCTION

There are many ubiquitous applications that require high positioning accuracy (e.g. cm or mm). Plenty of systems, like GRIPS (Generic Radio based Indoor Positioning System) [5] concentrate on integration of wireless positioning methods (i.e. Bluetooth, WLAN), But they are usually either expensive to setup or their accuracy are higher than what is required. In case of fast deployment of such systems without prerequisite of convenient infrastructure one needs something straightforward. Thus Bluetooth access points were adequate for infrastructure.
Our criteria preclude the use of conventional Global Positioning System (poor indoor signal reception).There are other options as beacons (access points) such as IR, RF, and ultrasound tag, however, they are quite costly and cannot easily communicate with Pocket PC.
The prototype manipulates unique ID of Bluetooth device. Therefore, it is possible to dedicate an ID to each room (proximity of 5 m). The costly part of this method is numerous Bluetooth beacons that are required for each room. Fortunately, cheap Bluetooth devices compensate expenses of numerous Bluetooth devices needed here. The deployment of application is simple and satisfies the needs of accuracy of positioning and navigation.

## 2.POSITIONING

It is necessary to create a distinction between position and location. The position of a device may be provided as 44.154º latitude and -94.0235º longitude, whereas the location of the device may be provided as two blocks south and three blocks west of the high school [3]. It is also possible to provide location via schematic plans. To be specific, this paper deals with the location of device. Navigation part also doesn't receive any coordinate from user but the name of location (a node of network) it wants to achieve.

## 3.SHORTEST PATH

Graphs are mathematical abstractions that are useful for solving many types of problems in computer science. There are many shortest path algorithms like 'Bellman-Ford' with Time complexity of O(N*E), where N and E are the number of nodes and edges respectively [2]. 'BFS' (Breadth-first search) time complexity is O(N+E)-not for weighted graphs- and generically, 'Dijkstra' has a complexity of $O(N^2)$. Dijkstra is able to calculate the shortest path from a single node to all the other nodes in the network [1]. A*, the algorithm used here is goal-directed version of Dijkstra and it finds shortest path, meaning that it aims to find the shortest path from the starting node to a particular destination (goal) [1]. A* is extensively used in path finding and graph traversal. A* plays a prominent role for its performance and accuracy.
A* performance is due to the fact that it is heuristics. It uses a best-first[1] search and finds the shortest path between two particular nodes.
A* uses a heuristic function (f(x)) composed of two other function as follow [10]:

- the cost function, which is the cost from the starting node to the current node (usually denoted as $g(x)$),
- and an admissible "heuristic estimate" of the distance to the goal (usually denoted $h(x)$).

Therefore one can write: f(x) = g(x) + h(x) where h(x) must be admissible-heuristic, therefore, it must not overestimate the distance to the goal. h(x) may represents straight-line distance to the goal.
h (x) satisfies an extra condition of $h(x) \leq d(x, y) + h(y)$ for every edge x, y of the graph. It means that if the equation is not satisfied then an overestimated distance to goal (h(x)) of x node of graph has been obtained. In other word if it overestimates h(x) it is likely to ignore the path leading to the

[1] **Best-first search** is a search algorithm which explores a graph by expanding the most promising node chosen according to a specified rule [9].

shortest path and selecting a path which actually is not the shortest one. This may not happen because A* uses an "optimistic" estimate of remaining path which means the cost of a path from current node to the goal will be at least as great as the estimate. But A* algorithm ensures that optimistic estimate is achievable. It is the main reason that A* is guaranteed to find an answer.
The time complexity of A* depends on the heuristic. In the worst case it has exponential growth [10].
Since the codes have been programmed in Microsoft C#, one could take advantage of a sophisticated graph library named "Quick Graph" [11]. This library ported the Boost Graph Library (BGL) [12] from C++ to C#. Quick Graph managed to fully support the application analysis requirements. It implements the same algorithm [13] mentioned for path finding in graphs.

## 4.USE CASE MODEL

As a project manager one should provide an easy to understand schema for the customers. Requirements analyses are the key elements of any software projects. Speaking of requirements, everybody knows that the overwhelming majority of litigation regarding software projects is based on misunderstandings over requirements. A very vivid diagram of the implementation is necessary to be pointed out here. There exists two main use cases, finding the position and finding the destination route. Destination detection uses positioning use case which means device position presumably is known just before trying to find the path. The use case diagram of the setup is as shown in Figure 1.

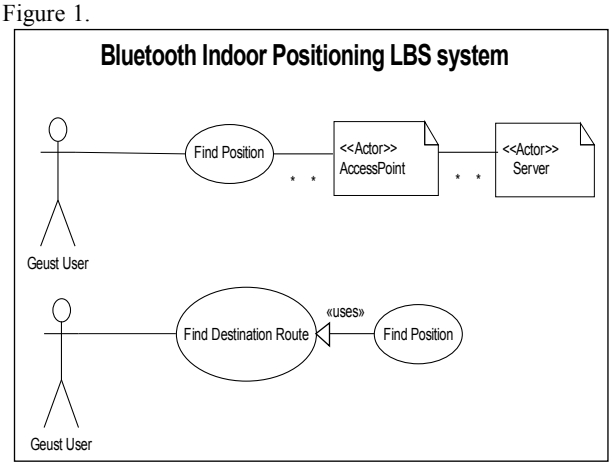


Figure 1: System use cases

And to clarify the details of a use case model, use case scenario are shown In Table 1.

## 5.EXPERIMENTAL SETUP

The implementation in this paper consists of three tiers as below [Figure 2]. For each of these tiers there is an application. There is a thin client with a straightforward application. Most of process carries out in the server. The access point acts as a link between the server and the device. Access points also cache Bluetooth IDs for a short period of time.

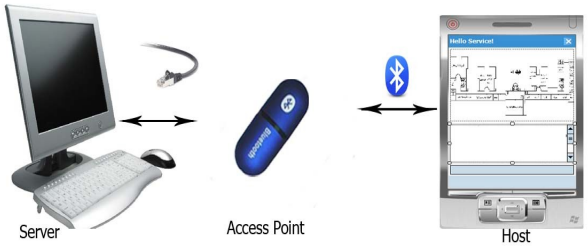


Figure 2: Three tiers architecture

### 5.1Guest (Device)

For all of Bluetooth programming procedure Microsoft stack .NET library called Inthehand [8] was utilized. The device is able to search the area via classes provided through the mentioned library. The area may be surrounded with a lot of Bluetooth devices, therefore, the search option should just list those beacons which have a specific name within them (like BT#--- beacons) and filter out other devices. Usually the first Bluetooth beacon appears on screen is the nearest one and the one to be selected.
The guest user then receives its location which is a map of its location and a brief description of the place. It also receives a list of places the guest can navigate through. After submitting the request to server, all process take place at the server and the result will be sent to the guest. Figure 3 shows thin client application at different states:


Figure 3: Thin-Client application

### 5.2Access Points

Access point is the middle tier and the key component [Figure 4]. It not only acts as a Bluetooth beacon (unique ID) but also as a bridge between the device and the server. It communicates with server via TCP/IP connection. It also receives request from the guest via Bluetooth. All of these transactions are logged, and any error can be tracked. Server's IP address or domain name can be entered by the user. It is also possible to hard-code domain name of server. Thus, if the IP of the server is changed, the network DNS resolves the server name.

Table 1: Use Case Scenario

Find Position use case narrative:

| Field Name | Field Description |
|---|---|
| Assumptions | Valid user and has permission to use this feature |
| Pre-conditions | User has an Windows Mobile with our thin client application |

| | |
|---|---|
| Use Case initiation | This Use Case starts on demand |
| Use Case dialog | … |
| Use Case termination | … |
| Post-conditions | Server logs Bluetooth ID and it's position |

| Field Name | Field Description |
|---|---|
| Use Case dialog | The user search area with its Bluetooth module |
| | Available access points list on user screen |
| | User selects first (nearest) access point |
| | A request will be sent to access point |
| | Access point send its Bluetooth id to server |
| | Server return respective access point information to the access point |
| | Access point send back the information to user |
| | User receives data and view on screen |

| Field Name | Field Description |
|---|---|
| Use Case termination | The user may cancel |
| | User search area and find no access point |
| | The Use Case may timeout |
| | The server doesn't respond to access point due to limitation of concurrent users. |
| | Network goes down during the process |

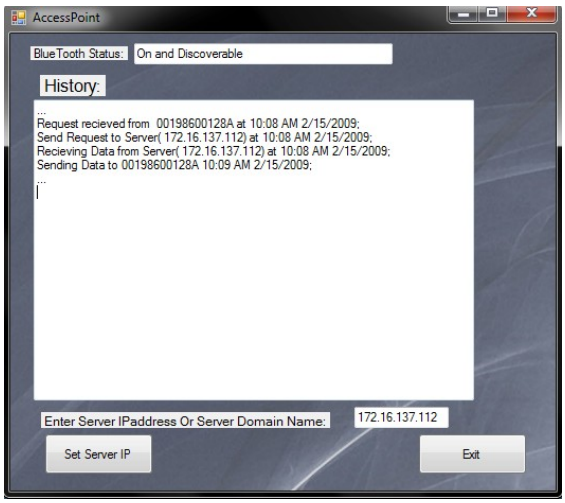

Figure 4: Access point application

### 5.3 Server

The main role of server beside spatial analysis is to receive access point address and check whether it exists in database. If it exists, then it queries the database and retrieves appropriate information. Server also keeps a log of device ID for a specific period of time and registers device current node (position) [Figure 5].

Figure 5: Server application

At the beginning of positioning process of a particular device, server queries the database and selects all available places (except the current position of device) the user can navigate to. ER (entity relational) diagram is used to design the conceptual model [Figure 6]. As it is shown in the figure, there exists three tables called Rooms, Points and Lines.

Points are defined as a primary key by combining three properties: X and Y coordinates and the floor number where the point belongs to.

Each graph node is related to a point from Points table and yet again every single room represents a point in Points table, meaning Point column in Rooms table is a foreign key associated with Point ID in Points table. Cardinality between Rooms and Points table is one to one. Lines table has three columns. Line ID is the primary key of this table and each unique line (ID) consists of two points (graph nodes) from Points table. Therefore the cardinality is two to one. These lines are considered as graph edges for future analysis such as shortest path finding.

There are different options for DBMS but MS Access satisfies the requirements here as there is no need to high level of security while it suffices the total number of simultaneous demands.

In large enterprises it's possible to migrate to other databases. For instance Oracle™ is an alternative which provides some handy tools (SQL Developer) facilitating migration from other DBMSs. Note that spatial extension of DBMSs like oracle is not required in this use case. It is again cost beneficial to just buy an Oracle Standard Edition for one CPU (Server) and no other oracle technologies is required to deliver Bluetooth services to a wide range of concurrent users.

A quick survey of a sample building is conducted and plans of floors are designed and drawn with AutoCAD and exported as a raster image (GIF).

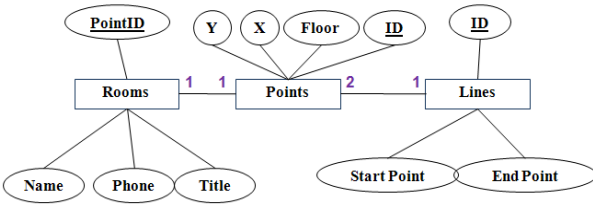

Figure 6: ER diagram of database

With a custom point digitizer, which is programmed for this purpose, graph (Nodes, Arcs) of all routes directly is led into the database [Figure 7]. It is an easy and handy way of populating database so fast. Points and Lines tables [Figure 6] have all essential graph data. By implementing A* algorithm

on the mentioned table, shortest path between any two given nodes is achieved. All the computation is done on server unobtrusively. The user device selects a place (node), its current node is known, and then the shortest path is calculated. The challenging part is navigation between distinct floors. In the case nodes are at different floors, first, shortest path from source to stairs is calculated and then again from stairs in other floor to the destination. In this case two different images are sent to users in which the paths in both floors are marked as red [Figure 8].

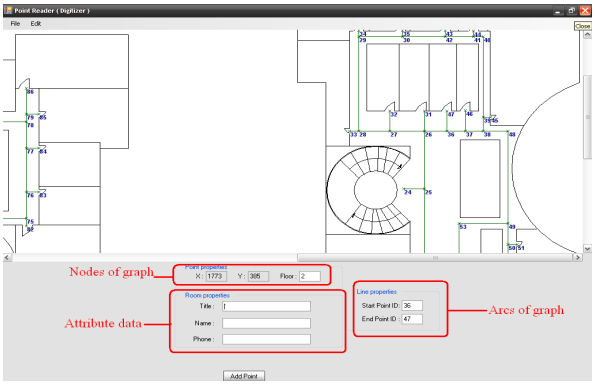


Figure 7: Custom Point Digitizer

Graph network has already been drawn on the maps in design phase. By opening GIF raster image into the "Point Digitizer" and then clicking on nodes of graph their image coordinates are captured and represented as specific properties of the point. Point floor, room's attributes should be manually set. Point ID also must be typed in "Line properties" part of the program. If the first field (Start Point ID) is occupied, all of other attributes will be regarded as its properties, else, meaning both of the fields are occupied then other attributes will be regarded as End Point ID properties. In the latter case, Lines table will also be populated.
Since it is an undirected graph it won't cause any problem to switch start and end point ID. In other words, the order of point selection is not a matter of importance.

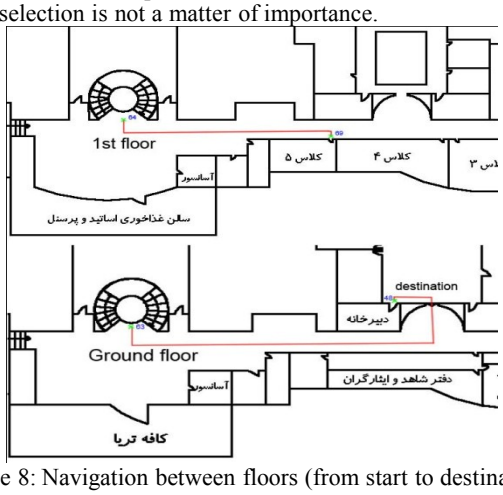


Figure 8: Navigation between floors (from start to destination)

## 6.SYSTEM PERFORMANCE

Obstacles such as walls and moving objects don't inhibit system functioning. It is enough to receive a signal even if it has a poor signal strength. Experiments show that Bluetooth signal made it through three 12cm width walls which is pretty promising. Current mobile device uses Bluetooth class 2 with a theoretical range of 10 m [6] which in practice is 5-6 m. Maximum distance between two Bluetooth beacons should be 10m to provide a good coverage. Number of requests to server doesn't exceed total number of access points, therefore buildings with 255 rooms are easy to manage with Access DBMS due to number of concurrent users to access DBMS is 255 [8].

## 7.ACKNOWLEDGEMENTS

It is necessary to thank University of Tehran Staffs for letting us implement the system over the Main building. A great appreciation goes to Mr. "Mostafa .Esmaeili" who helped us with floor plans and data (point digitizer).

## 8.REFERENCES

### 8.1References from Books:

1) Worboys and Duckham (2004), GIS: A Computing Perspective, Second Edition, CRC Press

2) Ronald L. Rardin (1998) , Optimization in operations research

### 8.2References from Other Literature:

3) Kiran Thapa,An Indoor Positioning Service for Bluetooth Ad Hoc Networks

4) Kenneth C. Cheung, Stephen S. Intille, and Kent Larson "An Inexpensive Bluetooth-Based Indoor Positioning Hack" Cambridge, MA 02142-1605 USA

5) T. Magedanz1, F. Schreiner2, H. Ziemek3, "GRIPS Generic Radio based Indoor Positioning System"

### 8.3References from websites:

6) Bluetooth Basics , http://en.wikipedia.org/wiki/Bluetooth

7) Access 2007 specifications, http://office.microsoft.com/en-us/access/HA100307391033.aspx

8) InTheHand .Net libraries for Microsoft Bluetooth Stack. Available at http://www.32feet.net.

9) Best-first search, http://en.wikipedia.org/wiki/Best-first_search

10) A* search algorithm, http://en.wikipedia.org/wiki/A*_search_algorithm

11) QuickGraph library, http://quickgraph.codeplex.com/

12) Boost Graph Library (BGL), http://www.boost.org/doc/libs/1_43_0/libs/graph/doc/index.html

13) QuickGraph, Graph Data Structures And Algorithms for .Net, http://quickgraph.codeplex.com/wikipage?title=Shortest%20Path&referringTitle=Documentation