University of Crete
Computer Science Department

VISITORS' MONITORING IN A MOBILE LOCATION-AWARE INFORMATION SYSTEM

by

STEFANOS DOUMBOULAKIS

MASTER'S THESIS

Heraklion, November 2008

University of Crete
Computer Science Department

VISITORS' MONITORING IN A MOBILE, LOCATION-AWARE, INFORMATION SYSTEM

by

STEFANOS DOUMBOULAKIS

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Author: _____
Stefanos Doumboulakis

Board of enquiry:

Supervisor _____
Constantine Stephanidis, Professor

Supervisor _____
Anthony Savidis, Associate Professor

Member _____
Yannis Tzitzikas, Assistant Professor

Approved by: _____
Panos Trahanias, Professor,
Chairman of the Graduate Studies Committee

Heraklion, November 2008

Abstract

A new generation of information delivery systems is emerging, known as location-aware systems. These systems have the ability to locate the user spatially and to automatically deliver relevant information through mobile devices —laptops, PDAs or cell phones. Therefore, these systems produce an environment of "enhanced-reality," where "on-the-go" information about areas and items of interest is provided with little or no user effort. In this context, the subject of this Thesis is a mobile location-aware information system, with particular focus on its monitoring component and positioning information delivery system. The above system components were developed with the following two key objectives: (a) to support efficient and speedy positioning information delivery in large-scale setups with very crowded use sessions at the scale of hundreds of simultaneous visitors, and (b) to enable real-time monitoring of all visitors from varying remote locations with 3D graphical representation.

The design and implementation of the information system's monitoring component has been carried out addressing the following issues: (a) the implementation of a monitoring client application, displaying the visitors in their surroundings as they move around, and the relevant tools for searching, alert control and virtual navigation; (b) the implementation of a monitoring server application responsible for gathering positioning information from location-sensing engines and exchanging such information with the hand-held devices, logging important information for future inspection, cooperating with another instance of itself to ensure robust, fail-proof operation and carrying out device activation requests; (c) the implementation of a networking library to enable wireless network disconnection detection and guarantee message integrity and atomicity; and, (d) the implementation of drivers and APIs for handling novel location-sensing technologies.

Περίληψη

Μία νέα γενιά πληροφοριακών συστημάτων αναδεικνύεται, γνωστά ως συστήματα με επίγνωση θέσης. Αυτά τα συστήματα έχουν την ικανότητα να γνωρίζουν τη θέση του χρήστη στο χώρο και να προσφέρουν αυτόματα σχετικές πληροφορίες μέσω κινητών συσκευών όπως είναι οι φορητοί υπολογιστές, τα κινητά τηλέφωνα και τα PDA. Έτσι αυτά τα συστήματα παράγουν ένα περιβάλλον «βελτιωμένης πραγματικότητας» όπου πληροφορίες για περιοχές και σημεία ενδιαφέροντος παρέχονται με ελάχιστη ή καθόλου προσπάθεια από τον χρήστη, καθώς αυτός κινείται. Σε αυτό το πλαίσιο, το θέμα της παρούσας εργασίας είναι ένα κινητό πληροφοριακό σύστημα με επίγνωση θέσης, με επίκεντρο το υποσύστημα παρακολούθησης και το μηχανισμό μεταφοράς πληροφορίας θέσεως. Τα παραπάνω συστατικά στοιχεία του συστήματος υλοποιήθηκαν με βάση τους παρακάτω δύο σημαντικούς στόχους: (α) την υποστήριξη αποτελεσματικής και ταχέως μεταφοράς πληροφορίας θέσης σε ευρείας κλίμακας εγκαταστάσεις του συστήματος με πολυπληθείς χρήστες της τάξης εκατοντάδων ταυτόχρονων επισκεπτών, (β) την παρακολούθηση σε πραγματικό χρόνο όλων των επισκεπτών από διάφορες απομακρυσμένες τοποθεσίας με τρισδιάστατη γραφική αναπαράσταση.

Ο σχεδιασμός και η υλοποίηση του υποσυστήματος παρακολούθησης έγινε με στόχο την αντιμετώπιση των παρακάτω θεμάτων: (α) την υλοποίηση μίας εφαρμογής-πελάτη για την παρακολούθηση, που εμφανίζει τους επισκέπτες και τον περιβάλλοντα χώρο τους καθώς κινούνται, με τα σχετικά εργαλεία για αναζήτηση, έλεγχο συναγερμών και εικονική πλοήγηση, (β) την υλοποίηση του εξυπηρετητή παρακολούθησης που είναι υπεύθυνος για τη συλλογή πληροφοριών θέσης από τις μηχανές εύρεσης θέσης και την ανταλλαγή τους με τις φορητές συσκευές, για την καταγραφή συμβάντων του συστήματος, για τη συνεργασία με ένα άλλο στιγμιότυπο του εξυπηρετητή που εξασφαλίζει την αδιάκοπη λειτουργία του συστήματος, και για την πραγματοποίηση των αιτήσεων ενεργοποίησης συσκευών, (γ) την υλοποίηση ενός υποσυστήματος δικτύου για τον εντοπισμό αποσυνδέσεων στο ασύρματο δίκτυο και την εξασφάλιση της ακεραιότητας και ατομικότητας των μηνυμάτων, και (δ) την ανάπτυξη λογισμικού για το χειρισμό νέων τεχνολογιών εύρεσης θέσης.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους επόπτες της μεταπτυχιακής μου εργασίας Αντώνιο Σαββίδη και Κωνσταντίνο Στεφανίδη για την συνεχή καθοδήγηση και υποστήριξή τους τα τελευταία τεσσεράμισι χρόνια στο πλαίσιο της συνεργασίας μας στο Εργαστήριο Επικοινωνίας Ανθρώπου-Υπολογιστή, του Ινστιτούτου Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας.

Επίσης, θα ήθελα να ευχαριστήσω τους Μανώλη Ζηδιανάκη και Νίκο Καζέπη με τους οποίους είχα τη χαρά να συνεργάζομαι συνολικά δύο χρόνια στο πλαίσιο της εκπόνησης των μεταπτυχιακών μας εργασιών που είχαν ως κοινό στόχο την ανάπτυξη του παρόντος συστήματος.

Table of Contents

List of figures

List of Tables

1. Introduction

Mobile location-aware information systems are designed to deliver to a portable hand-held device content information associated to the device's position, while the user roams freely the areas covered by the location tracking systems.

In the system reported in this Thesis, multiple location-sensing technologies are employed ([12]), offering both explicit positioning that deliver two-dimensional coordinates such as the Global Positioning System and a wireless location-tracking engine, and implicit positioning as the result of close proximity to a sensory module, be it an infrared or a radio beacon. The simultaneous deployment of different location tracking solution results in, sometimes, delivering conflicting positioning information, a problem solved by statically prioritizing all solutions according to their precision.

In such a mobile system, it is necessary to monitor all portable devices in order to have an immediate, clear and real-time view of both the system's condition and each device's functional state and location. Obviously, another important task is the delivery of positioning information from the various location-tracking systems to the portable devices, as well as to the monitoring component of the system. Activation information has also to be delivered to various components of the system. It concerns the initiation and termination of the user's session holding the portable device. Activation and deactivation requests are initiated by a desktop application, designed to allow charging a fee for a user session, and are propagated to the devices via the monitoring system in order to complete the activation process. Finally, the monitoring system also allows for the control of a device's state and virtual location in order to facilitate testing the system's deployment in a specific setup, as spatial features, such as obstacles, multiple floors and building materials, greatly affect the location-tracking systems' performance.

1.1 The Overall System

The overall system's key features regarding content administration, user navigation and runtime management of user sessions include: (a) location-triggered information delivery and on-demand content exploration by the user, (b) multiple location-sensing technologies deploying WLAN, GPS, FM and Infrared beacons, (c) single infrastructure for both indoor and outdoor setups, (d) efficient device renting facilities through the use of barcode readers, (e) spatial data editing with graphical editor for both mobile, on-site administration of location data and non-mobile, off-site administration of semantic content.

The system utilizes the Pocket-loox n560 PDA by Fujitsu-Siemens as the mobile device helping the users navigate in the host environment. This device features an Intel PXA270 processor, 64 MB of RAM, a 3.5 inch VGA 640x480 screen, Wi-Fi chipset, IR port, GPS module, a 2 GB SD-card and its operating system is Windows Mobile 5. The navigator application of the system makes use of the above PDA subsystems in order to receive location-sensing information, communicate with the rest of the system's application via wireless network and present location-triggered information to the user. Content information is stored locally using both a database and the filesystem. Content can be updated automatically at the end of the user session via network, whenever the administrator of the content management subsystem chooses to update. Also, at the end of the user session, location history data are propagated to a statistics server, over the network, and stored to a database. Apart from the navigator application, there are two more applications, one for auto-updating the navigator application and another one for registering the device with the system. Each PDA has a barcode tag that is a unique identifier and facilitates renting by using barcode readers.

The content management subsystem is used for associating semantic content information, in the form of text, audio, images and video, with items of interest located in the host environment. Additionally, one can create and edit areas and items of interest, create and edit navigation scenarios and define a map's boundaries. A Microsoft SQL server hosts the system's database, while an FTP server is also used to

store and distribute content information. This subsystem essentially provides the ability to add, edit, structure and delete content data and update obsolete data found on the devices. A much more simplified version of this content authoring application exists, which runs on the PDA, and that is the mobile content administrator. It is used for associating content with locations of interest, but this is done on-site with the location-sensing systems active and feeding the mobile administrator with positioning information while the user is on the go.

Several desktop applications control the processes of registering PDAs, renting PDAs, and keeping statistics. In order to restrict access to the system and store device identification information, each PDA's IP address, MAC address and barcode is registered using a server application that gets the above data via network from the PDAs and stores them to the database. The Renting application is used to initiate a rent or unrent process by using a barcode reader to scan the PDA, much like the way it is done in super-markets for any product. During the renting process, the type of user session is determined, whether it is multimedia or simply auditory, the language of the user session is set and the nationality of the user is recorded. A statistics server gathers location history data from PDAs via wireless network, whenever a user session is terminated and stores that information to a database.

1.2 Objectives

The developed monitoring system, besides providing a display of an area's overview and the location of each device, was developed with the intent to offer a fast and reliable solution for information delivery, tools for device searching, virtual navigation, alert control, and finally to provide a truly robust, distributed system with multiple instances of its client and server components running on different machines and cooperating simultaneously. The monitoring system's features are:

- Displaying visitors in their surroundings (map overview of the greater area and its boundaries, items of interest, etc) in 3D to facilitate scrolling and zooming, as they change position in real-time.

- Displaying information about a device such as MAC, IP address, barcode, location-tracking mode, user session information, state (connected / disconnected, on alert), etc, and offering the possibility to perform a device search with the above criteria.
- Maintaining a network connection using TCP/IP to each activated device, by detecting disconnections (no network coverage, wireless roaming) and auto-reconnecting as soon as possible.
- Receiving positioning information from the devices (actual position or change in location tracking mode) and processing it.
- Providing to the devices positioning information acquired periodically from a Wireless LAN location-sensing engine, and processing this information to identify new devices and position changes.
- Triggering boundary alerts and transmitting them to the devices when visitors step out of the map's boundaries.
- Providing activation information to the devices, which initiate or terminate the user session. The activation process is conducted in cooperation with the devices and the Activator application.
- Virtual navigation for the devices. The tool's user can explicitly position a device anywhere on the map, on any map. The user can also control a device's alert status explicitly.
- Logging important data to keep track of the system's operation and identify scenarios where problems / malfunctions appear.
- Robust monitoring-system operation that overcomes hardware and software failures by simultaneously deploying multiple cooperating server instances.
- Remote monitoring using the server-client model. Ability to run the GUI client portion of the tool in a remote machine while accessing the information in the server as it is running.
- Multiple GUI clients connect to the same Monitoring server, so the system can be simultaneously monitored from different machines / locations.

1.3 Architecture

The monitoring software tool comprises three applications: the Dynamic PDA Distribution Server, the Monitoring Server and the Monitoring GUI Client. The Distribution Server gathers positioning information from the wireless location tracking engine and propagates it to the Monitoring Server. The Monitoring Server has network connections to the PDAs, exchanging activation and positioning information. The PDAs receive positioning information from GPS, infrared and FM beacons, which is sent to the Monitoring Server informing it about the PDA's current location and location-tracking mode. The Activator (aka Renting Application) sends activation requests to the Monitoring Server and the Server propagates the request to the respective device, waiting for a confirmation message. Successful activations and deactivations cause the active instance of the Monitoring Server to notify the backup instance of these events in order for both instances to maintain the same state. The purpose of the Monitoring Server's backup instance is to take over all monitoring operations when the active instance goes down. All the above information is also fed to the Monitoring Clients, by the Server, in order to keep their displays up-to-date.

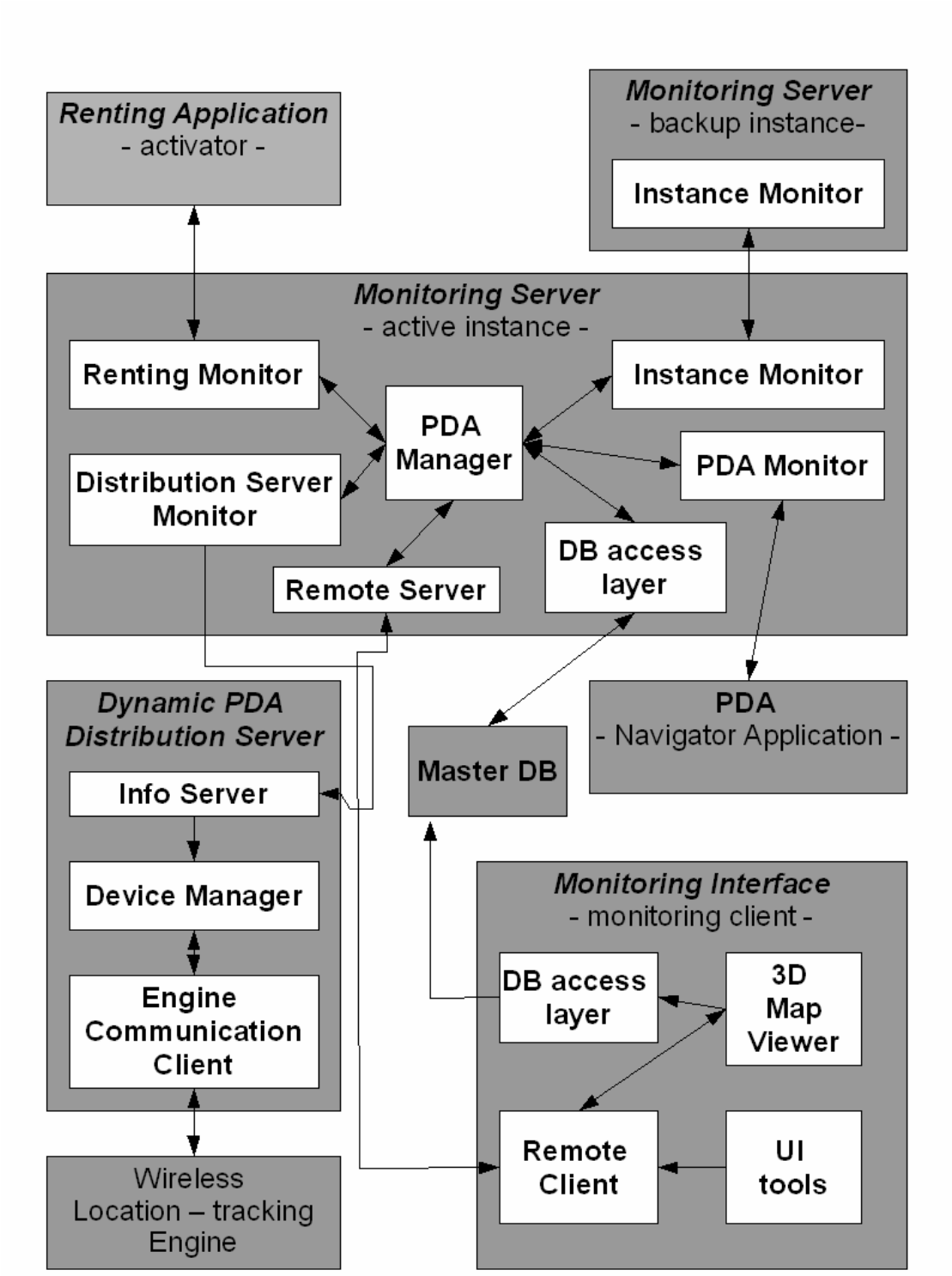

Figure 1 - Monitoring component architecture. Shaded rectangles represent applications while white rectangles represent components

1.4 Methodology

The implementation of a mobile location-aware information system constitutes a very large development effort. Therefore, the choice of the programming libraries and techniques was very important for the successful completion of the project within the bounds of the desired time-frame. The core development language was decided to be C#, because it is a modern object-oriented language with access to the .NET framework libraries and available DirectX bindings. The .NET and DirectX libraries are also available for the PDA, so it made sense to use C# and these libraries system-wide. DirectX was mainly used for 3D visualization in applications that demanded this feature, such as the Monitoring Client. The rest of the GUIs are .NET forms. The .NET Remoting API covered the need for RPC inter-process communication between the Monitoring Client and the Server. Because of the parallel development process, missing system components had to be replaced by simulation applications to enable proper testing of the Monitoring component. Simulation applications were developed to substitute the navigator application on the PDA and the Renting (activator) application on the desktop.

1.4.1 Testing and Benchmarks

In order to test the system while using hundreds of devices, a special application was developed to simulate the presence of these devices, as there were not enough devices available nor was it easy or efficient to conduct such a test with real devices. This application had one thread per simulated device running the PDA's navigator application monitoring manager, the component responsible for communicating with the Monitoring Server. Some modifications were made to the Monitoring Server, in order to produce fake location updates for all devices at fixed time intervals and deliver them to the application simulating these devices.

Running both applications on the same machine ensured that there would be no malfunctions due to heavy network traffic or poor wireless signal reception, so the software implementation of the application logic could be tested. Even with hundreds

of devices, all moving at the same time every second, there were no failed connections or any other kind of malfunctions. Running the two applications on different machines connected via wireless network with heavy traffic, in order to test the system in bad conditions, caused several network disconnections to occur. The number of these disconnections was proportionate to the number of devices in the system as expected. The failed connections were re-established as soon as the devices were accepting them again, as they were supposed to, and data delivered through the network arrived instantly. The most stressful test was done by running the simulation application on a machine with relatively poor wireless connection and moving five hundred devices all at the same time every second. This test lasted a couple of hours.

Another part of the monitoring component that was tested at the same time, while conducting these benchmarks, is the monitoring interface application. This application receives location updates from the Monitoring Server and displays the devices on a map. Increasing the number of devices and the rate of location updates would cause the frame rate of the monitoring interface to drop proportionately. Delivering a hundred more location updates every three seconds would cause the frame rate to drop an extra 35 frames per second. With a single device in the system, a frame rate of about 180 fps was observed, while 500 devices, all moving at the same time, caused the frame rate to reach 11 fps. At 11 fps, there is an obvious distortion in the smoothness of the visual display and the UI becomes much less responsive to user input. As a conclusion, it can be said that the number of location updates per second determine the frame rate, rather than the number of actual devices in the system. These benchmarks were conducted using a machine with specifications (Intel Core 2 Duo 6400, 1 GB RAM, Ati X1650) close to those of the actual PC to be used.

2. Related work

Location-aware information systems could not exist without the availability of location-sensing technologies. Some technologies rely on specialized, expensive and usually inflexible hardware installations such as the older Active Badge location system ([17]) and the Active Bat system ([4]), employing signal distance measurements to determine a user's position. C-MAP ([14]) was among the first exhibition-tour systems exploiting the Active Badge System for location awareness. Other systems based on distance signal measurements are the Cricket Location Support System ([11]), House of Matilda ([5]) and EasyLiving ([6]), the latter using vision methods via motion-tracking cameras.

Many mobile location-based information systems exist, relying on different position or context sensing technologies. Geographical Information Systems (GIS) are the most well-known systems on top of GPS technology, mostly suited for macro-scale map-based navigation and way finding, supporting coarse-grained information tagging over maps rather than micro-scale exploration with positioning and tagging at the level of distinct proximate exhibits. Outdoor information systems lack generality (i.e., they can not be used indoors), practicality (satellite signal can be lost) and precision at the level of information points (coarse-grained positioning). On the other side, indoor systems employ solely tag technologies, such as infra-red beacons or RFID tags, suffering from various limits. For instance, infra-red lacks position precision (conic signal diffusion), while RF id tags require users to be able to identify and touch specific areas near exhibits, as in MoVIS ([13]). Such systems essentially implement context-sensing rather than positioning, meaning user monitoring and tracking, or trajectory recording, as the developed platform, is not possible.

2.1 Active Badge

The Active Badge is a system for the location of people in an office environment. Members of staff wear badges that transmit signals providing information about their location to a centralized location service, through a network of sensors. An 'Active

Badge' is a hardware tag that emits a unique code for approximately a tenth of a second every 15 seconds (a beacon). These periodic signals are picked up by a network of sensors placed around the host building. A master station, also connected to the network, polls the sensors for badge 'sightings', processes the data, and then makes it available to clients that may display it in a useful visual form. The system features a location server that is responsible for collecting data from sensors, processing them and displaying them on screen. The monitoring capabilities include searching for a badge, searching for badges in the vicinity of a certain badge, listing the badges of a specific location, an on-sight notification mechanism and displaying the location history of a badge. The on-screen display of the server is a textual representation of the list of badges, their current location, and the probability of a badge being in that location.

2.2 Active Bat

In the Active Bat system a matrix of ultrasonic receivers is placed in the rooms' ceiling tiles, acting as beacons, while tags – dubbed 'Bats' – are attached to users and equipment. A radio transmitter polls Bats in turn triggering ultrasonic emission while ceiling receivers measure time from radio poll to ultrasonic reception to calculate distance. At this point, a tag's position can be computed using the calculated distances from the ceiling receivers. Each Bat has a unique identity that can be used to enforce physical location security, such as access control to certain parts of the environment and safety requirements regarding the number of people allowed inside a specific area. The Active Bat system provides monitoring by displaying 2D graphical representations of users and maps of their surrounding areas. User-defined spatial zones are also displayed on the map. The system can also display the location history of a user and the relative time spent on each location by employing different colours when drawing each spot on the map.

2.3 Cricket

Cricket consists of location beacons that are attached to the ceiling of a building, and receivers, called listeners, attached to devices that need location. Each beacon periodically transmits its location information through an RF message. At the same time, the beacon also transmits an ultrasonic pulse. The listeners listen to beacon transmissions and measure distances to nearby beacons, and use these distances to compute their own locations. This active-beacon passive-listener architecture is scalable with respect to the number of users, and enables applications that preserve user privacy. The Cricket system is essentially a decentralized location support system that aims to achieve user privacy and decentralized administration by not advertising location information of a user to a central server or to its peers, and therefore does not have a monitoring application.

2.4 EasyLiving

EasyLiving is a prototype system for building intelligent environments. It is not merely a single "intelligent room", but rather a software toolkit with which intelligent environments can be constructed. The system's key features are: (a) computer vision software to track people and maintain each person's identity, (b) a geometric model database that records information about objects and people in the world, (c) a thumbprint scanner to establish people's identities, (d) an event system and scripting system that can trigger actions based on people's movement and personal scripts that tailor the actions of the environment for each individual. Demonstration programs include: (a) playing each person's pre-selected movies when they sit in front of display screens, and stopping them when they stand up; (b) playing a game of "Hotter and Colder" as a person searches for the goal spot in the room; and, (c) redirecting the output from a wireless mouse to whichever computer screen is closest, as the person holding the mouse moves from place to place in the room. The EasyLiving system also lacks central monitoring.

2.5 House of Matilda

House of Matilda is a project aimed at creating smart environments to enable elderly persons to live a longer and more independent life at home. Ultrasonic sensor technology provides an indoor precision tracking system that allows the smart home to make proactive decisions to serve its occupants based on context-awareness. An OSGi-based framework abstracts the ultrasonic technology into a standard service to enable the development of third-party tracking based applications. This system enables remote monitoring by allowing applications to register with a location service which periodically sends location updates. The monitoring application graphically displays the user and a map of the immediate surrounding area.

2.6 Ekahau Positioning Engine

More recent location sensing systems rely on standard wireless networking hardware, by measuring radio frequency (RF) signal intensity and attenuation to determine a user's location. The RADAR system ([1]) was amongst the first tracking systems based on IEEE 802.1. A commercial system in this category is the Ekahau Positioning Engine (EPE) ([2]) supporting laptops, PDAs and other Wi-Fi enabled devices, accomplishing floor, room, and sometimes door-level accuracy, while working indoors and outdoors. In the developed platform, EPE was deployed as one of the point-based location sensing technologies (i.e., it is not the only one used). The official specifications of EPE indicate that it is capable to reach an average precision of 1 meter, though it was observed that in real practice the average precision is around 3 meters.

EPE provides monitoring capabilities as web services and also features a web interface for server control and device monitoring. The web services can be used to deliver device listings and positioning information filtered by using predefined criteria such as location accuracy, quality, etc. The software tool presented in this thesis makes use of these web services. The server's web interface provides a list of all devices registered with the server, meaning that they have been used at least once. The user can choose to see the graphical representation of a device in its surroundings

given its last known position. There are two shortcomings in this representation approach. Firstly, the user can only see a single device and not all the devices present in a certain area, and second, the display is not updated automatically, forcing the user to continually refresh the web page.

Table 1 - Comparing alternative location sensing technologies

	Manual	Infrared	RFID short	RFID long	GPS	WLAN
Cost	*Zero*	*Low*	*Low*	*Medium*	*Low*	*Very high*
Line of sight	*No*	*Yes+*	*No*	*No*	*Yes-*	*No*
Accuracy	*On spot*	*2-10m*	*5-25cm*	*3-5m*	*~2m*	*3+ m*
Indoors	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*Yes*
Outdoors	*Yes*	*Low*	*Low*	*Low*	*Yes*	*Yes, extra h/w*
Energy	*Zero*	*Low*	*Low*	*Medium*	*Low*	*High*
Usability	*Minimal*	*Low*	*Very low*	*Average*	*Great*	*Great*

2.7 Comparison

Table 2 summarizes the aforementioned monitoring traits of the featured location-aware systems.

Table 2 - Comparing systems' monitoring features

	Active Badge	Active Bat	Cricket / EasyLiving	House of Matilda	Ekahau	Thesis System
Monitoring UI	Text	Graphical 2D	-	Graphical 2D	Text / graphical 2D	Graphical 2D / 3D
Search Tools	Yes	No	-	Not applicable	Yes	Yes
User-defined Zones	No	Yes	-	No	No	Yes
Location History	Yes	Yes	-	No	No	No
Remote Monitoring	No	No	-	Yes	Yes	Yes

Comparing the evaluated systems with each other and the system presented in this thesis, there are a few things worth noting. While Active Badge and Ekahau have mostly textual representation in their monitoring UI, most systems employ a 2D graphical representation. Also, most systems feature a search tool, except for Active Bat and House of Matilda, which is a single-user system and does not require such functionality. The monitoring UI, of both Active Bat and this thesis' system, has the ability to display user-defined zones on the map depicting some area covered by the system. The Ekahau system also features user-defined zones but they are not displayed on its monitoring interface. The Active Bat and Active Badge systems are capable of presenting the location history of their users on the monitoring interface, while this thesis' system does not display such information although it is recorded and can be a viewed using another component of the system that does statistical analysis. Regarding remote monitoring, House of Matilda, Ekahau and this thesis' system feature services delivering location updates that third-party applications can register with. Also, the monitoring UI of Ekahau and of the system discussed in this thesis, can be deployed remotely since it follows the server-client model.

3. Networking Library

The need for integrity and atomicity of the system's network traffic data and the detection of network disconnections due to wireless roaming has led to the development of a networking library. Such library consists primarily of wrapper classes of the .NET Socket class, representing the entities of the classic server-client model of TCP-IP networking, and a class responsible for serializing and deserializing data into messages. A custom text-based protocol is used for simplicity and ease of debugging network messages.

3.1 Message Integrity and Atomicity

To ensure integrity, the data to be sent are packaged using the format described below and they are accordingly verified upon reception. The header of the message consists of a fixed-sized string indicating the total size of the message in characters, and a single-character flag indicating whether it is a user or a system message. The body of message is the string that needs to be sent. When a client receives data, it inspects the header to determine the size of the actual message and verify its proper reception. A properly received message will cause the client to fire a “Data Received” event, delivering the message to all interested parties. The concurrent arrival of multiple messages triggers an equal number of “Data Received” events, while incomplete data are buffered and no event is fired until the rest of the message is received. Thus, message atomicity is also achieved, meaning that a single send of data will trigger exactly one event upon reception that will deliver only this data.

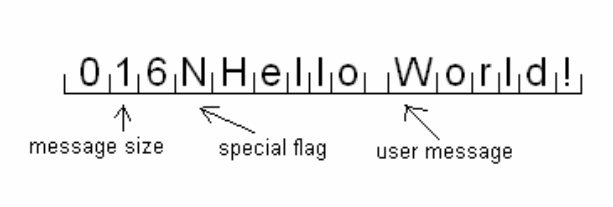

Figure 2 - “Hello World!” encoded in the message exchange protocol

3.2 Detecting network disconnections

A client is able to detect a network failure by means of sending and receiving special small messages at regular intervals, whose failure to arrive in time indicate a lost connection. When a socket connection is established, the client who initiated the connection starts sending periodically a “ping” message to its peer, so that the peer will know whether the connection is alive. At the same time, the client checks if a similar “pong” message has arrived to determine connectivity. The client who accepted the connection acts similarly by checking for received “ping” messages at double time interval, but sends “pong” messages as soon as a “ping” message arrives. When data other than these special messages arrive at a client, the connection is considered alive as if a special message has arrived within this time period even though it may have not. If a client fails to receive a special message from its peer within the set time period, the connection is considered dead and is automatically closed. It is important to choose carefully the time period, because a long one will make failure detection slower, while a short one will cause alive connections to shut down if a peer has heavy workload and fails to send its special messages or if there is heavy traffic in the network that stalls communication.

4. Monitoring Server

The Monitoring Server keeps track of all devices in the system, whether activated or not, as well as of their location and status. It is responsible for determining and delivering such information to the devices themselves and to the Monitoring Clients. It also gets device positioning information from the location-sensing engine and the devices, performs activation requests from the Renting Application, and communicates with its cloned instance to ensure fail-safety.

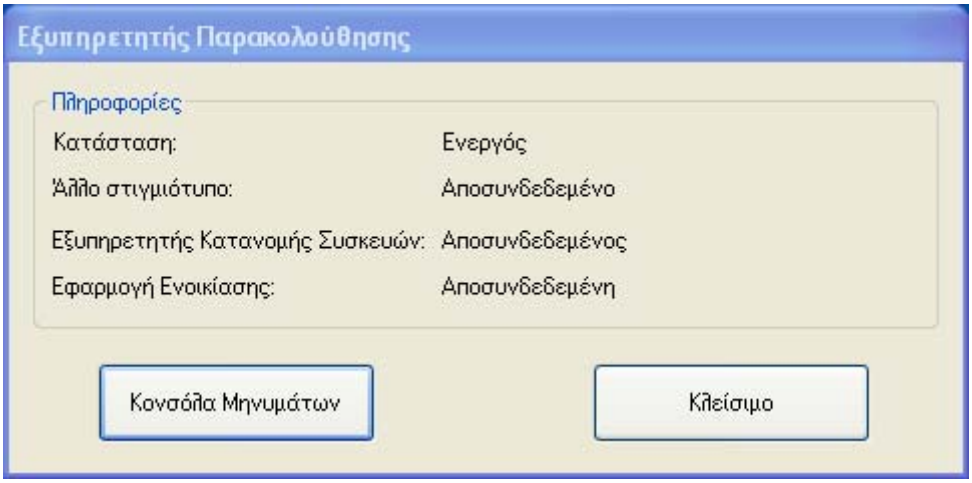

Figure 3 - The Monitoring Server UI

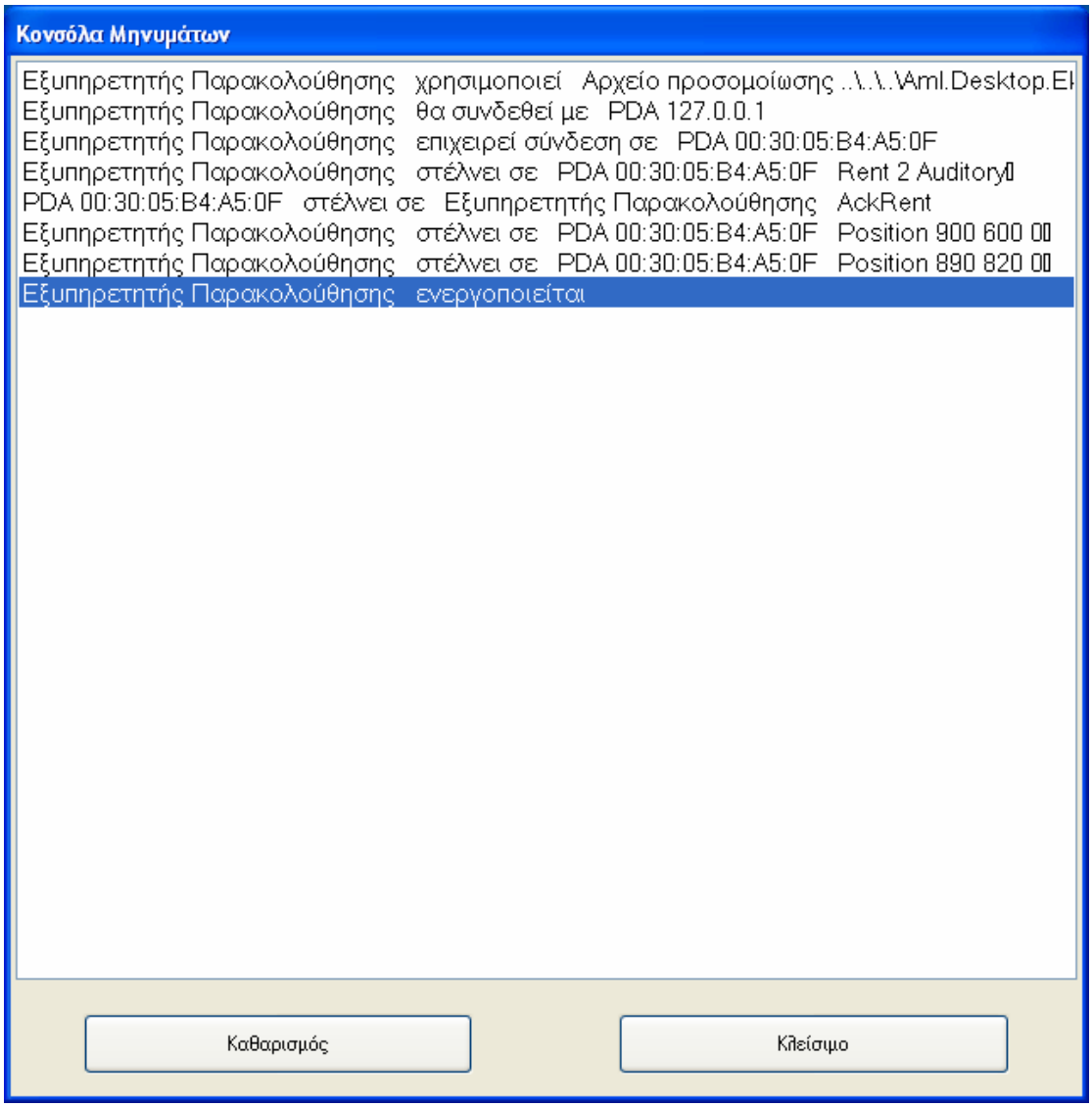

Figure 4 - The Monitoring Server's message console

4.1 Location-Sensing Engine Communication

The Dynamic PDA Distribution Server retrieves positioning information from the Wireless LAN location-sensing engine, processes it and sends it to its clients, mainly the Monitoring Server. It periodically requests the positions of all devices from the location-sensing engine by using the engine's web services. Upon the engine's response and based on information from previous requests, it determines whether there are new devices or if some devices have been removed, and also calculates changes in positions. The timely arrival of the engine's response to a request is used to

determine the engine's responsiveness and this information is displayed on the server's GUI. The processed positioning information is packaged using a custom text-based protocol and passed on to the clients at regular intervals. Multiple clients may connect to a single instance of this server. The Server's GUI also displays errors occurring in clients' connections and the data received from the engine, as well as the data delivered to the clients. The Server also has the ability to log location information coming from the positioning engine in order to facilitate debugging and assessing the engine's performance.

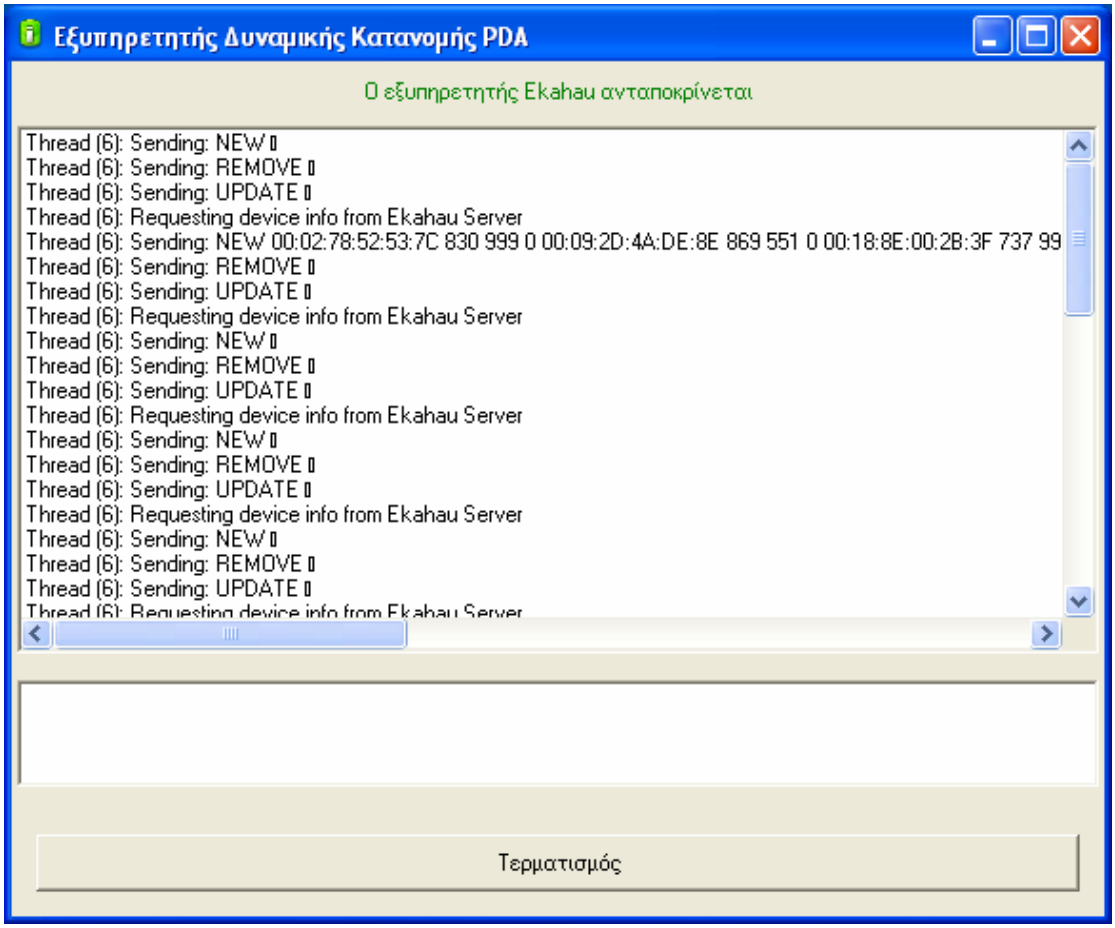

Figure 5 - The Dynamic PDA Distribution Server UI

4.2 PDA Communication

The Monitoring Server is responsible for delivering and receiving positioning and activation information to and from the PDAs. This is achieved by maintaining TCP-IP network connections. Such a connection may fail if a PDA is located in an area with poor wireless network coverage, or if the PDA is performing wireless roaming from one access point to another. The Server detects connection failures using the functionality offered by the Networking Library and continuously attempts to reconnect to disconnected PDAs.

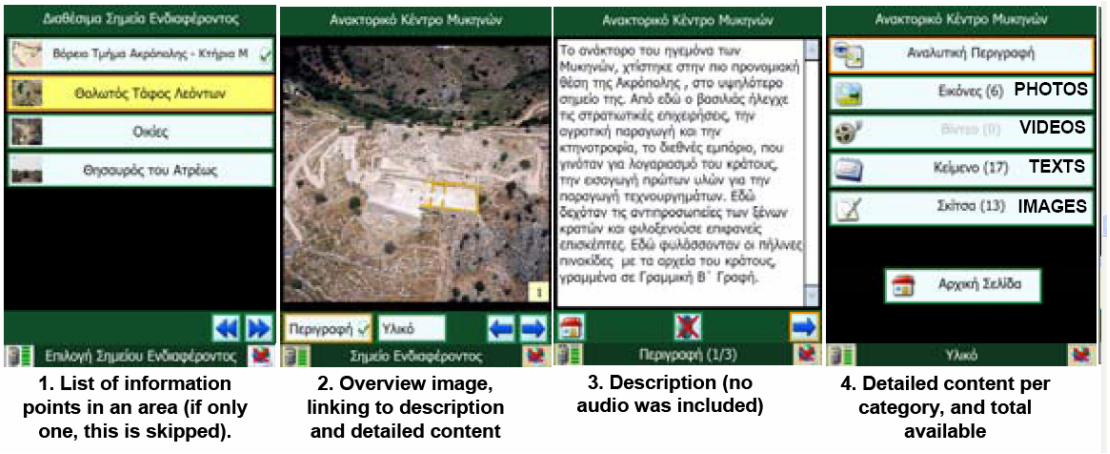

Figure 6- Views of the mobile Navigation System

4.2.1 Activation & Deactivation

The Monitoring Server is connected via network to the Renting application, which is responsible for initiating the rent / unrent (activate / deactivate) procedures, which, if successful, will initiate / terminate the PDA user's session. An activation notification is a message delivered from the Renting Application via network to the server and forwarded to the PDA. An acknowledgement message is sent back by the PDA and forwarded to the Renting Application in order to conclude the procedure. Failure to send the activation message to the PDA or to receive the acknowledgement is reported to the Renting Application, so that it is known why the renting process failed.

A similar path of action is followed for deactivation, but even if no acknowledgement message is received the PDA is considered deactivated. The Monitoring Server is capable of accepting connections from multiple Renting Applications and treats them separately, meaning that a Renting Application only gets information about PDAs that it has activated itself.

4.2.2 Positioning Information Exchange

While a PDA is activated, it constantly receives positioning information from the Monitoring Server, but it also sends back positioning information if it is navigating by using GPS or infrared beacons. This way, the PDA has all the available information to optimally determine its position and the Monitoring Server is kept up-to-date regarding the PDA's current position and location tracking mode. This information is also sent to the Monitoring Client(s) to keep it up-to-date. Every time a PDA moves to a new position, the Server performs a check to determine whether the PDA is inbounds and if not, a boundary alert is raised which is forwarded to the PDA and the Monitoring Client. Likewise, whenever a PDA moves back within map bounds, a similar notification is sent.

4.3 Server Redundancy

In order to survive hardware or software failures, two instances of the server have to run in two different machines. Only one of the two instances is active, meaning that it performs whatever functions it is supposed to perform. The other instance is simply waiting until it has to take over when the active instance fails. The two instances are connected via network, so they can detect each other’s crashes when the connection fails and they exchange state information, notifying each other as to whether they are active or inactive, and forwarding activation / deactivation notifications. The inactive instance is always connected to a Dynamic PDA Distribution Server and receives the same positioning information as the active instance.

The two instances are connected by a TCP-IP socket connection. One instance gets to be the server and the other the client, which is determined by a configuration option. When a connection is established, the client sends a message to the server indicating its active or non active status. If the client is active, then the server is deactivated if necessary. If the client is inactive, then the server is activated. The instance that was already active sends a list of all activated PDAs to the other instance, so that it will know what has already happened. In order to be kept up-to-date, the inactive instance also receives an activation or deactivation message whenever a PDA is successfully activated or deactivated by the active instance. When the inactive instance detects that the connection with the active instance is broken, it is activated and attempts to connect to all PDAs listed as active (rented). If such a connection is successful, the PDA sends back its position and its location tracking mode. If the connection fails, reconnection attempts are made continuously until they succeed. At this point, the previously inactive instance of the Monitoring Server is active and operating normally.

4.4 Monitoring Client Communication

The Monitoring Client communicates with the Monitoring Server via network by performing remote procedure calls (RPC) using the .NET Remoting API. The Server exposes, through a library that both the Server and the Client link to, the necessary functionality, by offering lightweight wrapper classes as remote objects. These wrapper classes implement the required functionality through delegate calls to the actual heavyweight classes' functions. Many instances of these lightweight classes may be created simultaneously to serve independently multiple clients at the same time. In addition to performing explicit RPC calls, the Client may also declare interest to remote events fired by the Server and have local callback functions invoked. If the invocation of a callback function fails, the respective Client connection is considered broken, and all remote objects created by this Client are destroyed.

```
public abstract class CommonRemote : MarshalByRefObject
{
    protected String m_hostName;

    public String HostName
    {
        get { return m_hostName; }
        set { m_hostName = value; }
    }


    public override object InitializeLifetimeService()
    {
        return null;    // infinite lifetime
    }


    public void Disable()
    {
        RemoveCallbacks();
        ClearEvents();
        RemotingServices.Disconnect(this);
    }


    protected abstract void SetCallbacks();
    protected abstract void RemoveCallbacks();
    protected abstract void ClearEvents();

}//class
```

Figure 7 - Remote objects inherit from this class

```
[Serializable]
public class SAOFactory : MarshalByRefObject
{
    private static Dictionary<Type, Type> m_types;

    public static event EventHandler<CreateRemoteEventArgs> OnCreate;


    static SAOFactory()
    {
        m_types = new Dictionary<Type, Type>();
    }


    public static Dictionary<Type, Type> DaTypes
    {
        get { return m_types; }
    }


    public T Create<T>(String hostName) where T : CommonRemote
    {
        Type derivedType = m_types[typeof(T)];

        T t = (T)Activator.CreateInstance(derivedType);

        t.HostName = hostName;

        if (OnCreate != null)
            OnCreate(null, new CreateRemoteEventArgs(t));

        return t;
    }

}//class
```

Figure 8 - The server factory creates remote objects

A remote object, specifically its server-side implementation, does not follow the same rules as normal objects regarding its lifetime and atomicity, because it does not appear in any block of code written by a developer. Thus, special care has to be taken to prevent it from being garbage-collected unexpectedly or not getting collected at all, and to ensure whether a single or multiple copies of the object will be created. To satisfy those requirements among others, the Server exposes the remote type information of a single remote object that acts as a factory for creating all the other remote objects. Every time a remote object is created through the aforementioned factory, its Client host name is passed as an argument to the “Create” function and the

newly created object is registered to be owned by the specified Client. Periodically, a check is performed to determine broken or terminated connections to Clients, and the respective remote objects are finalized and explicitly garbage-collected. Remote objects are set to infinite lifetime in order to prevent garbage collection and avoid the sponsor-lease system offered by .NET, which places a heavy burden on network traffic by regularly contacting all Clients requesting sponsorship for all remote objects.

The Client may move from one instance of the Monitoring Server to the other one when the active instance fails or the client is connected by accident to the inactive instance. This is accomplished by checking periodically server connectivity, simply by attempting to perform a remote procedure call. Failure to perform the call means that the server is down, so attempts are made to connect to the other instance. The same happens when any other RPC call fails. At client startup, after a successful connection to the server, the server is queried as to whether it is active. On a negative response, the client breaks the connection and attempts to connect to the other instance.

4.5 Logging

The Monitoring Server has logging functionality in order to keep track of the system's operation and help the user identify scenarios where problems appear. Every significant action is logged with a time stamp, such as information exchange via network regarding device positioning and activation, server redundancy and network connection / disconnection events. All this information is in human-readable form and it also appears on the Server's UI message console. Some aspects of the logging functionality are configurable by the user, specifically the number of log files, the number of entries per log file and the number of log entries that will cause the flushing of the log to the file. At startup, the Server examines the log files and, based on the above settings, determines the current log file. If all log files are full, then the first log file gets to be the current one and its contents are deleted. When the current one is filled then the next one is deleted and recreated.

Figure 9- A sample log file

5. Monitoring Client

The Monitoring Client is a 3D application depicting the areas covered by the location-tracking systems and the activated devices located in these areas. Static information such as maps, areas of interest and exhibit items are downloaded from the system's database at startup, while dynamic information such as list of devices and their state and location is delivered by the Monitoring Server. This application has several display options and tools to facilitate the monitoring and control of the devices. The UI's text may be in Greek or in English as with all applications of the system.

5.1 Main Display

The main window of the application consists of the 3D map overview, the main menu and camera toolbar on top, and the display options and information on the bottom of the screen as well as the right edge of the screen. Using the main menu, one can change the map being displayed, bring up a tool's window or exit the application.

5.1.1 Map Overview

One map is always displayed, that takes up the largest portion of the screen, with virtual areas and items of interest, the map boundaries and the activated devices located on that map. Areas of interest are polygons drawn on the map that group several items of interest which are located in proximity of each another. Items of interest are represented by blue dots on the map pinpointing their location. A red polygon represents the map's boundaries, which determine whether a device steps in or out of bounds. This is necessary because the map overview of an area may contain extra space for reasons of symmetry. Activated devices, located on the current map, are displayed as circles whose color varies depending on the device's state, which can be normal, disconnected or on alert. The circle denotes a device's location and it

moves around as positioning information arrives from the Monitoring Server. A device can be selected by clicking on its circle or by iterating the devices through the camera toolbar. Overlapping circles can be selected by multiple clicks that iterate through these circles. When a device is selected, a big tooltip appears next to it that contains information regarding that device such as its MAC address, IP address, barcode, tracking mode, state, session type, session language and activation time. The above information, as well as the circle's color, change dynamically according to new information regarding the device's state, coming from the Monitoring Server. The user can configure the following settings: the circle's size, the tooltip's size and the circle's colors.

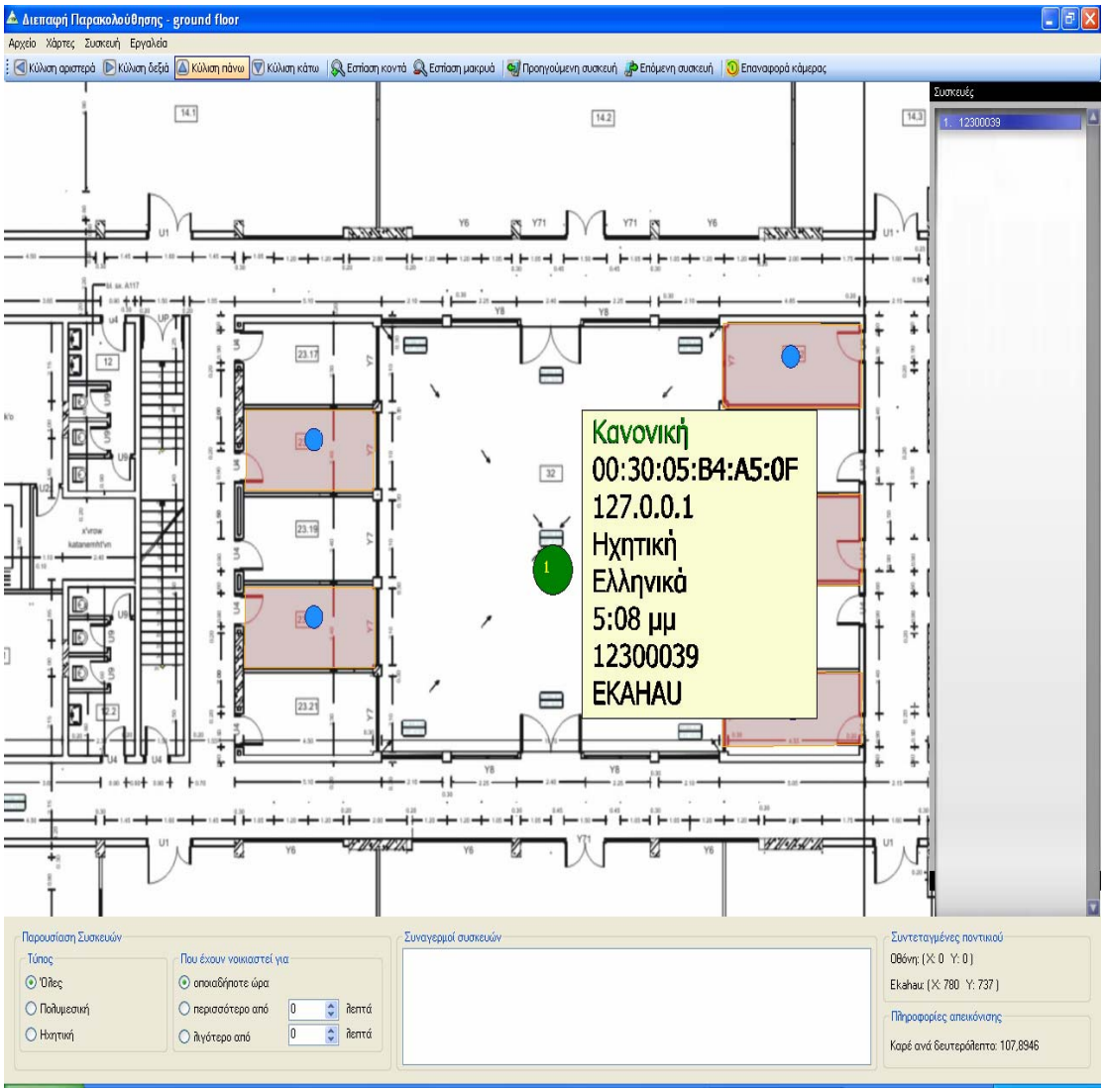

Figure 10- Monitoring Client main display

5.1.2 Camera

The camera looking at the 3D environment can be moved around either explicitly using the mouse or implicitly by auto-focusing on the selected device. Some parameters regarding the camera can be configured by the user, such as the speed of auto-focusing, the minimum height that auto-focusing will reach, the scroll speed, the manual zoom speed, and whether manual zoom focuses on center of the screen or on the mouse pointer. The mouse wheel controls zooming, while a toolbar is available for all camera operations such as scrolling, zooming, resetting and focusing on each device. The above operations also have keyboard bindings. Zoom and scroll speed are proportional to the camera's height, so as to maintain smoothness and sensible speed. The camera always auto-focuses smoothly on the selected device, unless a device is not selected.

5.1.3 Display Options and Information

The right and bottom edge of the screen are taken up by windows with display options regarding the devices and information about the display and boundary alerts. A big list box is found at the right edge that lists all devices located on the current map. Clicking on a device from that list, the device is selected as it would have been selected clicking on its circle on the map. On the bottom window, the user can select which devices are displayed according to the following criteria: session type and elapsed time since activation. Right next to that, a list is displayed of the boundary alerts that occurred, marking the device's location when it went off bounds. Clicking on a device on this list will select that device if it is activated. On the bottom far right edge of the window the mouse position coordinates are shown both in bitmap and DirectX numbering, as along with the frames per second displayed by the application.

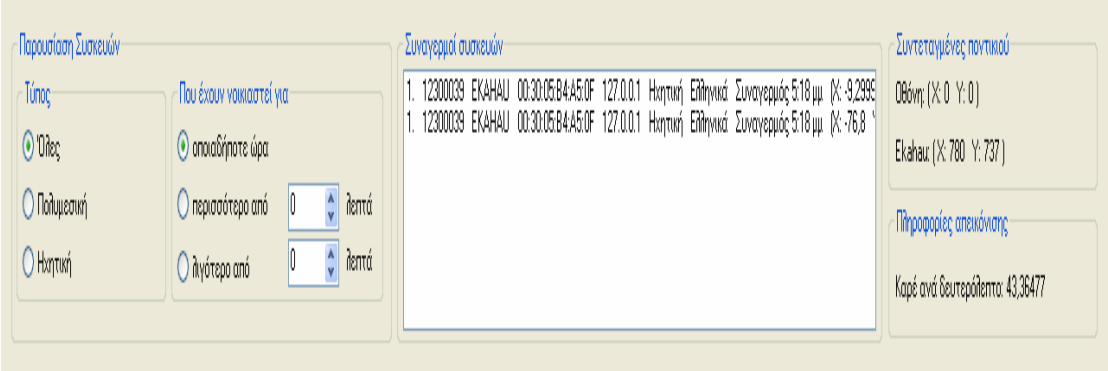

Figure 11 - Display options and information

5.2 Tools

All tools have a window of their own. The window is not modal, so it can remain open. This also facilitates selecting a different device to act upon, by clicking on it in the map display without having to close and reopen the window. The devices listed in these tools are updated dynamically, meaning that a change in their state will be reflected in their representation (color and text description change) and they will also be removed from the list if they get deactivated. A device's text description is colored according to its state, allowing easier identification. Selecting a device from a tool listing will also cause the same device to be selected in the map overview, changing the map if necessary and focusing the camera on this device.

5.2.1 Alert Control

This tool facilitates the inspection and manipulation of the devices' state, especially whether they are on boundary alert or not. This tool's window has a tab notebook, where each tab corresponds to a map and has a list of the devices currently located in that map. A device may be selected from the list and hitting the “Alert” button will set it on or off alert depending on its current state. The currently selected tab will automatically change if the user changes the map in the main window. Devices will automatically move from one tab to another when their location changes. This manual

alert control acts complementary to the automatic process of calculating if a device is off bounds, which is carried out by the Monitoring Server. The Monitoring Server is notified of the user-provoked boundary alert status changes, so it updates its own information and notifies the respective device and the other Monitoring Clients.

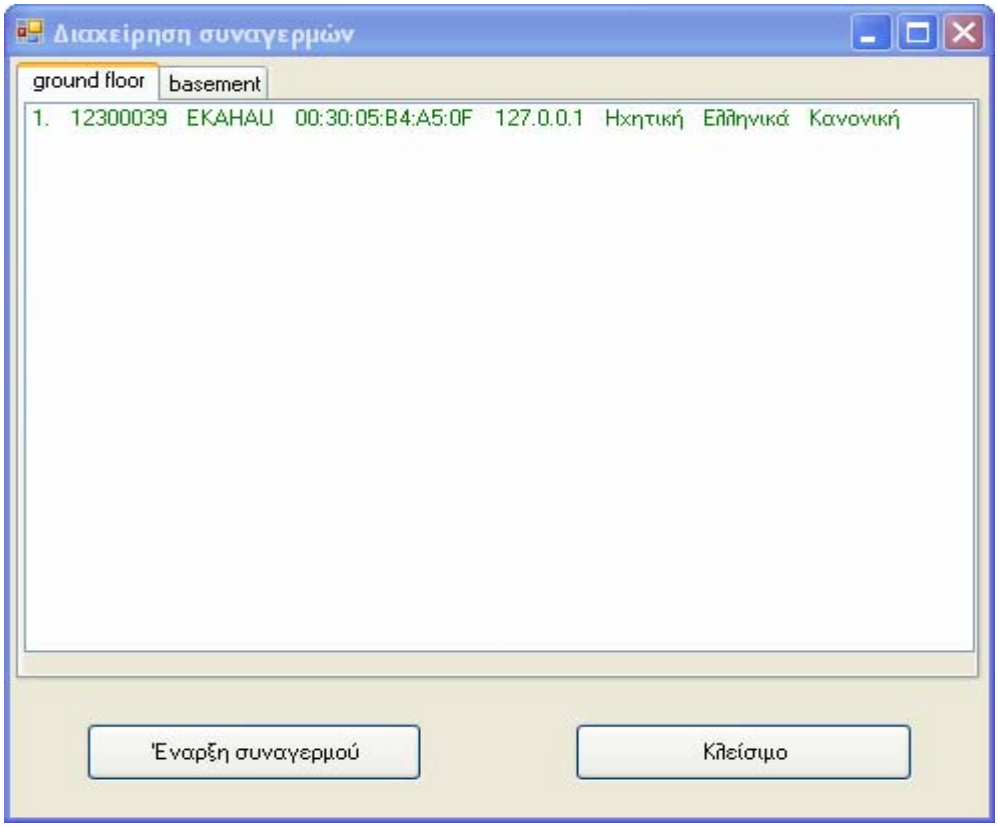

Figure 12 - Alert control window

5.2.2 Searching

By using the Search tool in the main menu, the user is able to perform a search for a single device or a group of devices that match the specified criteria. These criteria include barcode, MAC address, IP address, tracking mode, session type, session language, state and map. Predefined values are set to “any”, so hitting the “Find” button will fill the results' list with all activated devices. Criteria represented by a text field such as MAC, IP and barcode will cause a match against prefixes of the

respective device characteristics. The search results appear in a list box in the form of a list of device text descriptions and their total number is printed above this list.

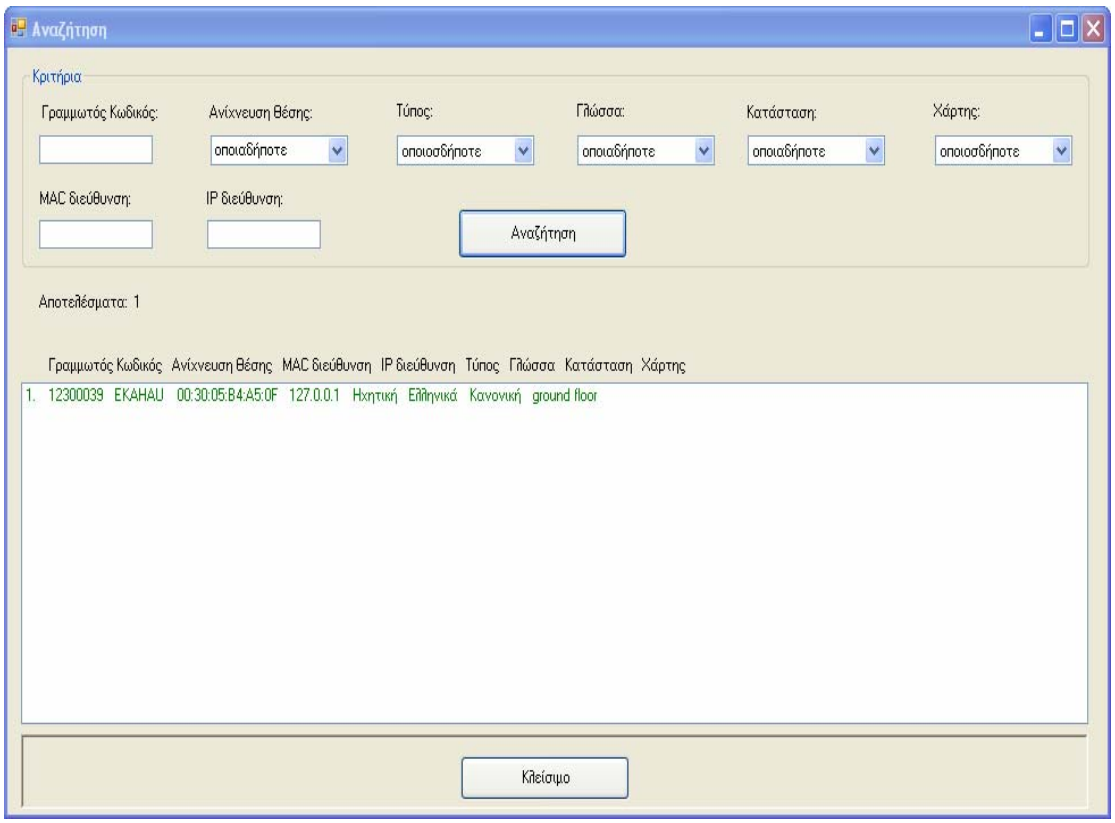

Figure 13 - Search tool

5.2.3 Device Removal

The Client's user has the ability to remove a device, so that it is considered as deactivated, under certain conditions. When a device is disconnected for a long period of time or if some other malfunction in the device's operation occurs such that its user session has effectively ended, the Client's user will need its removal. This is achieved by selecting a disconnected device on the map and selecting the relative tool from the main menu. The removal may only occur if the selected device is in the disconnected state. Upon removal, the Monitoring Server is notified, sets the device as deactivated and performs whatever actions are necessary.

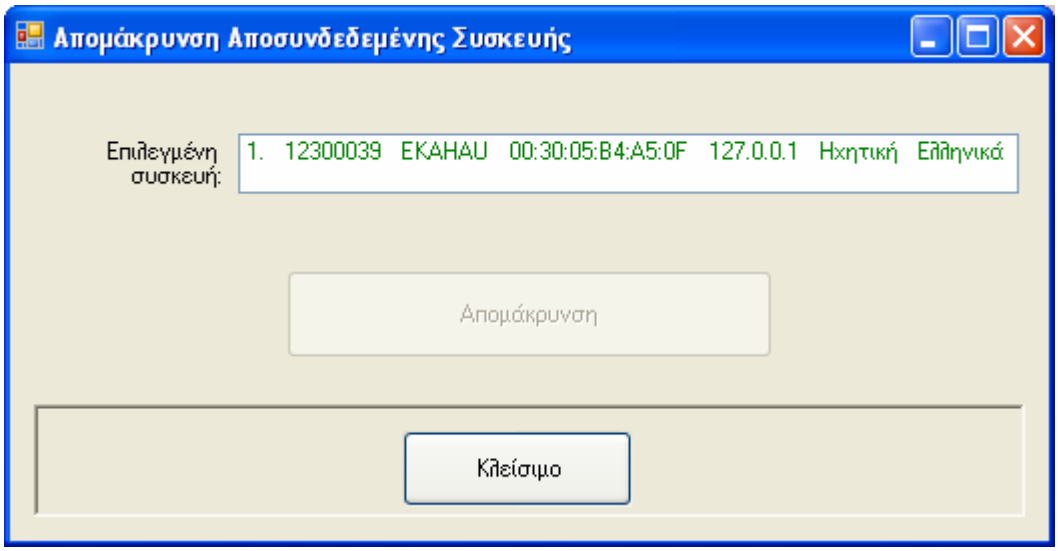

Figure 14 - Device removal tool

5.2.4 Virtual Navigation

For testing purposes, a device may be placed, by the user, anywhere on any map. Virtual navigation is enabled by setting a configuration option to prevent accidental use. When enabled, the user can drag 'n' drop a device to the desired location. In order to place it on a different map, the related tool must be selected from the application's main menu which allows the currently selected device to be placed in the middle of the screen. The Monitoring Server is notified when a device is placed explicitly by the Client's user, and its position is updated as if this information came from the location-sensing engine so the usual actions occur, a boundary check is performed, and the device and all other Clients are informed of the new position.

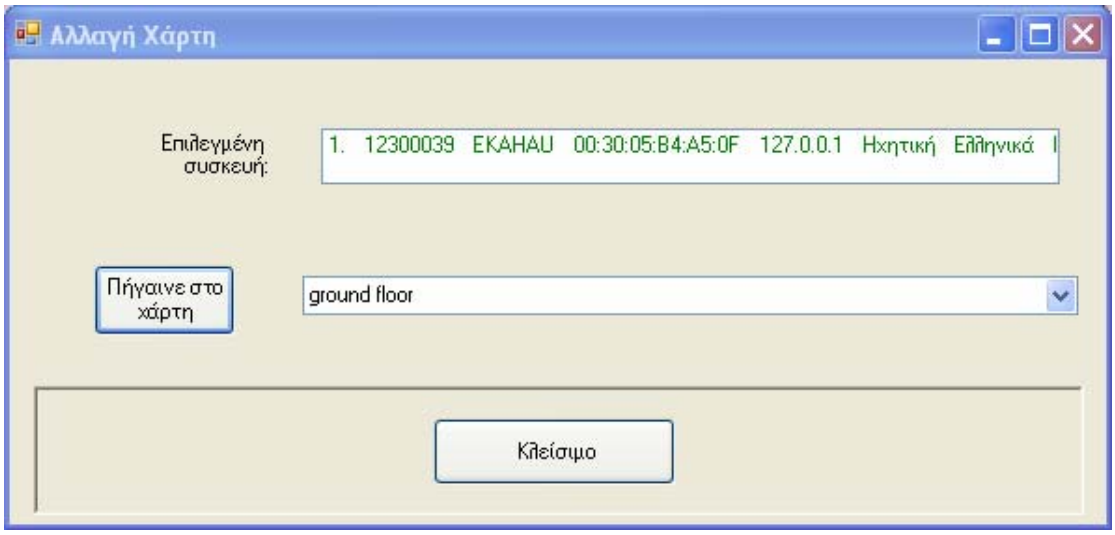

Figure 15 - Virtual navigation, map change

6. Location tracking

The system has four different means available for device location tracking, including GPS ([3]), a wireless location tracking engine, infrared beacons ([7]) and FM beacons ([15]), but not all of them are used in the same setup. Obviously, an exclusively indoor setup does not require GPS, while having both infrared and FM beacons is pointless as they both offer the same accuracy in close proximity. The wireless location-tracking solution is a system standard, especially for indoor areas, offering accuracy up to 2 meters, but not so much for outdoors because of the need to have multiple access points and the availability of GPS. A solution employing Ultra Wide Band ([16]) was considered, but it was discarded due to the strict requirements of placing the beacons in very specific locations, which is difficult or outright infeasible in most setups.

6.1 Wireless Location - Tracking Engine Setup

The Ekahau Positioning Engine (EPE) is the system's wireless location tracking engine, used primarily for indoor areas. Location tracking is performed via the EPE server receiving RSSI (Received Signal Strength Indication) measurements that PDAs get from access points. These measurements are processed according to a positioning model, created during the engine's calibration, and the PDAs' positions are calculated. The engine's calibration is essentially the association of many locations in the area with the respective RSSI from the access points, so when in real-time operation these or similar RSSIs are sent to the server, the server deduces that the PDA is in the location suggested by these measurements.

The EPE software consists of the EPE server, which runs on a server / desktop computer, the EPE client that runs on the PDA and the EPE calibration application that runs on a tablet PC. The client is responsible for getting the RSSI measurements and sending them to the server, while the server performs the processing. Using the calibration application, a positioning model is created by providing the maps of the respective areas, defining possible paths and locations of interest on those maps, and

finally walking along these paths and marking one's location while the tablet's wireless network interface card receives RSSI from access points. The application features a built-in EPE server that allows tracking of the tablet PC, and therefore the immediate testing of the calibration process. Unfortunately, actual real-time results using the PDA will vary from the tablet's results, because they have different wireless chipsets and RSSI translation might not be perfect. Another problem is the PDA's wireless chipset ability to scan for RSSI often enough in order to meet the demands of the system. A frequency of three or more seconds between scans introduces an obvious delay in the system when depicting the PDA's location.

The careful setup of the access points, covering the desired areas, and the calibration of the engine, will make the difference between poor and excellent location tracking. At least four access points are required to properly cover an area, while adding more access points, especially up to eight, will help yield optimal results. It is also good practice to use only non-overlapping channels of the wireless spectrum and to spread these amongst the access points, so that neighboring ones will not use the same channel. A proper calibration requires multiple survey points to be set rather than paths, because of the EPE server's tendency to place a PDA on those points. Thus, it is required to actually replace paths with a multitude of points, making the whole process slower and more strenuous. Each new version of the EPE would alter location tracking performance, so testing, experimenting with settings and recalibrating had to be done anew.

Figure 16 - The EPE Web Interface

6.2 FM - Infrared Setup

To fulfill the need for high accuracy in location tracking, especially when items of interest are very close to each other, FM or Infrared beacons are placed next to these items. Such a beacon is constantly transmitting an identification number within a virtual cone whose exact bounds are determined by the beacon's power and orientation. This helps the PDA distinguish between several items of interest that are

in proximity of each other, and it also complements the EPE's accuracy which is not sufficient when the PDA gets close to the items.

6.2.1 Infrared Beacons

Infrared beacons are easier to use because the PDA has a standard built-in IR port and therefore no extra hardware is required. On the other, hand infrared communication requires visual contact between the transmitter and the receiver which can be a problem, and it is a costly solution for large setups since there is a licensing fee per PDA on top of the cost of the beacons. These beacons can be powered by either battery or standard power outlet supply. A dynamic link library (dll) containing a COM (Component Object Model) object is provided by the manufacturing company that enables the programmatic control of the communication between the IR port and the beacon. The .NET environment automatically provides a C# interface for COM objects which made the integration of this library into our software trivial. Software provided by the manufacturer allows the configuration of the beacons' power, orientation, transmitted data, time interval between transmissions, etc. The manipulation of the aforementioned beacon settings allows to define virtual space zones where the PDA will pick up the IR signal of a single beacon, so conflicts between adjacent beacons can be avoided and the beacon's range can be essentially determined.

Figure 17 - Infrared beacons

6.2.2 FM Beacons

Because of the high cost of the simple IR beacons, another solution employing FM beacons has been developed by a company for the purposes of this system. This solution includes an FM transmitter, acting as a beacon, and an FM receiver, mounted on the PDA, with an IR transmitter that uses the PDA IR port as a bridge to achieve communication with the PDA. The extra hardware that has to be attached to the PDA is a drawback, but it helps avoid some issues that simple IR beacons have, such as visual contact and conflict resolution. Due to this product being a prototype, there was no driver software available, so it had to be developed inhouse. The IR port can be controlled using the standard socket library. Unfortunately, the .NET library's Socket class and IrDAClient class failed to work, so a small driver in C++ was developed for this purpose. The driver had to have very simple functionality as MFC timers and threads don't work very well when mixed and matched with .NET libraries and

applications. The driver is used in the software by means of the .NET P/Invoke mechanism, since the programs are written in C#. At driver startup, the IR port is queried for found devices and then a socket connection is established. When the FM receiver gets the '?' ASCII code through IR, it responds by sending back the FM beacon's id that has been recently received, if any. Future improvements are aimed at turning the receiver on and off programmatically, so as to conserve battery power.

7. Summary and Conclusions

7.1 Summary

This Thesis has presented the monitoring component of a mobile location-aware information system. The component's architecture has been presented along with its list of features. Subsequently, the functionality offered by the Networking library that allows disconnection detection and guarantees message integrity and atomicity has been discussed. Additionally, particular features of the Monitoring Server have been presented, such as the information exchange procedures, the server's redundancy, the logging process and the RPC communication with the Clients. Finally, the Monitoring Client and its tools have been presented, as well as the solutions adopted for issues raised by the employment of the various location tracking technologies.

7.2 Conclusions

The choice to merge the monitoring component of the system with the dynamic information delivery mechanism stems from the fact that the monitoring component had to be aware of this information so it made sense to be the one circulating it. Otherwise, there would have been a duplication of the same delivery procedures and a waste of system resources. This merge resulted in the monitoring component being split in two applications according to the client-server model.

The server part, responsible for information delivery, can run on server machines in remote places, constantly online, safely away from the monitoring client user. The client user is not necessarily the system administrator nor should he have physical access to the server, especially if his machine is close or exposed to the public. A typical setup of the system in a museum would include the monitoring client application running on a computer located in the museum's reception, and monitoring server would be running on a machine located in a server room away from the reception. Additionally, in large setups with multiple machines in the reception or

different reception sites, it is required that multiple users can monitor the system from different machines. Also, having at least two server machines means that the monitoring server can run on both of these machines simultaneously, guaranteeing uninterruptible operation despite hardware or software failure.

The client part of the monitoring component can run on a simple desktop machine without worrying about system failure. If the monitoring client goes down, no other part of the system is affected and it is a simple matter of restarting the client to get it up and running. It is also trivial to have multiple clients in different machines as already mentioned. Choosing 3D visualization for the monitoring client application gave the benefit of speedy zooming and scrolling functions and ease of implementing them. A future extension to the application could concern a “real” 3D representation of the areas, items and people carrying the devices, in contrast to the 2D overview and the flat shapes that are currently used.

Although the developed system already employs four different location-tracking technologies, it is possible that even more will be used in the future. Most of these technologies are complementary to each other, or cannot be used in all setups due to spatial and financial restrictions. Some of the location-sensing solutions being considered are the wireless tags for the EPE and the Ultra Wide Band system. The wireless tags are provided by the manufacturing company, and therefore have better precision than some other company's wireless chipset found inside a PDA. The UWB system was tested but discarded for museum setups due to spatial restrictions and bureaucratic reasons, but that does not mean it cannot be used in other projects / setups.

Future work may also include the addition of more alerts and notifications delivered to the PDA, as a way of paging users that wander in the host environment, from the location of the Monitoring client machine. This requires a slight expansion of the communication protocol, used by the monitoring component and the PDA’s navigator application, to include the new notifications. At the same time, additions and modifications should be made to both the Monitoring Client interface and the navigator application interface, so these notifications can be initiated and delivered to the user. The type of notification, whether aural, visual or both, will determine the

kind of UI implementation that is necessary. It may also be necessary to force some kind of identification of users, so the system will have knowledge of who is using each PDA.

Another possible improvement of the monitoring system is the addition of spatial information for each area, regarding walls, stairs, doors and other spatial features that restrict movement and limit pathways. The availability of such information is beneficial in two ways: (a) it allows to enforce safety and security policies such as limiting the number of people that may enter a room or prohibiting entrance to certain areas, and (b) it enables to improve the performance of location-sensing systems by ignoring location updates that don't make sense such as instantly covering great distances. The enforcement of security policies can be combined with the extra notifications, previously discussed, to inform violators of their actions and guide them in order to avoid further policy violations. To implement the aforementioned functionality, policy-related data and spatial information must be stored in the system's database and retrieved by the monitoring component, which will enforce these policies according to the location updates it receives from the various location-tracking systems. Alerts will have to be displayed on both the Monitoring Client and the PDA's navigator application, so the user interface of these applications will have to be modified accordingly.

In closing, it should be mentioned that the system presented in this thesis has been fully developed and successfully tested in real-world conditions, delivering on its promise to provide a set of authoring and management tools for a mobile, location-aware system, addressing information delivery in large-scale setups with crowded use sessions.

References

(1) Bahl, P., Padmanabhan, V.N. (2002). RADAR: An In-Building RF-Based User Location and Tracking System. In proceedings of IEEE INFOCOM 2002, Volume 2, pp 775-784.

(2) Ekahau Positioning Engine. http://www.ekahau.com .

(3) Global Positioning System. http://www.navcen.uscg.gov/gps/default.htm .

(4) Harle, R. K., Ward, A., Hopper, A. (2003). Single Reflection Spatial Voting: A Novel Method for Discovering Reflective Surfaces Using Indoor Positioning Systems. In proceedings of ACM MobiSys 2003, International Conference on Mobile Systems, Applications and Services, pp 1-14.

(5) Helal, S., Winkler, B., Lee, C., Kadoura, Y., Ran, L., Giraldo, C., Kuchibhotla, S., Mann, W. (2003). Enabling Location-Aware Pervasive Computing Applications for the Elderly.

(6) Krumm, J., Harris, S., Meyers, B., Brumitt, B., Hale, M., Shafer, S. (2000). Multi-Camera Multi-Person Tracking for EasyLiving. In Proceedings of the 3rd IEEE International Work-shop on Visual Surveillance (VS'2000), pp 3-3.

(7) Lesswire Infrared Beacons. http://www.lesswire.com .

(8) Microsoft .NET Framework. http://www.microsoft.com/NET .

(9) Microsoft COM. http://www.microsoft.com/com/default.mspx .

(10) Microsoft DirectX.

http://msdn.microsoft.com/en-us/directx/default.aspx .

(11) Priyantha, Nissanka B., Chakraborty, A., Balakrishnan, H. (2000). The Cricket location-support system. In proceedings of the ACM MOBICOM 2000, 6th International Conference on Mobile Computing and Networking, pp 32-43.

(12) Savidis, A., Zidianakis, M., Kazepis, N., Grammenos, D., Dubulakis, S., Stephanidis, C. An Integrated Platform for the Management of Mobile Location-Aware Information Systems. In proceedings of Pervasive 2008.

(13) Schwieren, J., Vossen, G. (2007). Implementing Physical Hyperlinks for Mobile Applications using RFID Tags. In 11th International IEEE Database Engineering and Applications Sym-posium (IDEAS 2007), Banff, Canada (Sept. 6-8), pp 154-162.

(14) Sumi, Y., Etani, T., Fels, S., Simonet, N., Kobayashi, K., Mase, K. (1998). C-MAP: Building a Context-Aware Mobile Assistant for Exhibition Tours. Community Computing and Support Systems, Springer LNCS Vol. 1519, pp 137 – 154.

(15) Symmetron FM radio. http://www.symmetron.gr .

(16) Ubisense Ultra-wideband. http://www.ubisense.net/content/14.html .

(17) Want, R., Hopper, A., Falcão, V., Gibbons, J. (1992). The active badge location system. ACM Transactions on Information Systems (TOIS), Volume 10, Issue 1 (January 1992), pp. 91-102.