Project no. 035003

u-2010

Ubiquitous IP-centric Government & Enterprise Next Generation Networks Vision 2010

Instrument: Integrated Project

Thematic Priority 2

D3.2.1 Report on the Presence Management Solution

Due date of deliverable: 31st August 2008

Submission date: 19th November 2008

Start date of project: May 1st 2006 Duration: 36 months

Organisation name of lead contractor for this deliverable: Lancaster University

Revision: v1.0

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract

This deliverable presents the design and implementation of a presence management solution for the emergency and crisis scenarios of u-2010. The requirements of the presence management solution are tailored for the Mountain Rescue scenario since this provides the most difficult set of requirements for a presence management solution to fulfil. However, the developed solution can be adapted for other emergency and crisis scenarios with particular suitability for search and rescue scenarios.

Keywords:

Mountain Rescue, Presence, Location, Management, GPS, Mapping.

History of Change

Issue	Status	Date	Details	Responsible
V0.1	Draft	27/06/08	General document skeleton and ToC.	Martin Dunmore
V0.2	Draft	28/06/08	Added Introduction	Martin Dunmore
V0.3	Draft	30/06/08	Added Requirements and LAS chapters	Martin Dunmore Panagiotis Georgopoulos
V0.4	Draft	01/07/08	Added Evaluation chapter	Martin Dunmore Panagiotis Georgopoulos
V0.5	Draft	18/07/08	Added LBS chapter	Martin Dunmore
v0.6	Draft	30/07/08	Updated requirements	Martin Dunmore
v0.7	Draft	10/09/08	Updated evaluation and LAS chapters	Martin Dunmore
v0.8	Draft	28/10/08	Updated LAS and LBS chapters	Martin Dunmore
v0.9	Final draft	31/10/08	Tidied up grammar, added executive summary, updated screenshots.	Martin Dunmore
v1.0	Final version	13/11/08	Minor changes in response to reviewer's comments	Martin Dunmore
v1.0	Final version	19/11/08	Finalization for sending to EC	Harold Linke

Table of Contents

List of Figures

List of Tables

Executive Summary

This deliverable presents the design and implementation of a presence management solution for the emergency and crisis scenarios of u-2010. The requirements of the presence management solution are tailored for the Mountain Rescue scenario since this provides the most difficult set of requirements for a presence management solution to fulfil. However, the developed solution can be adapted for other emergency and crisis scenarios with particular suitability for search and rescue scenarios.

The developed solution is called the Location Awareness System (LAS), which consists of mobile clients continuously updating a server (located at the rescue team's headquarters) with their locations. Location coordinates are obtained via GPS modules and are transmitted to the server using whatever the most suitable connectivity is available at the time. For the Mountain Rescue domain in the English Lake District, this generally consists of a choice between SMS, GPRS or a WiFi network provided by mobile routers. A connectivity framework has been designed to reflect the needs of the mountain rescue team we are working with in conjunction with the needs of the application. The client devices used are HP iPAQ PDAs and the client software runs under Windows Mobile. Theoretically, the client applications can use IPv4 and/or IPv6 to communicate although problems were noted using IPv6 under Windows Mobile. However, these problems do not prevent the software working correctly with future versions of Windows Mobile that have more robust IPv6 stacks.

Other problems attributed to Windows Mobile include an inability of the device to swap from GPRS to WiFi connectivity under certain conditions. However, despite these limitations, the LAS has been demonstrated to provide a comprehensive presence management solution for the mountain rescue scenario. Furthermore, the LAS components have been integrated into the larger solution for the Mountain Rescue scenario that is being carried out in WP4.

1. Introduction

This deliverable presents the design and implementation of a presence management solution for the emergency and crisis scenarios of u-2010.

The following sections in this chapter discuss the definition of presence management, the advantages a presence management solution can bring for emergency and crisis operations and how we see a presence management solution working in u-2010.

The rest of this deliverable is structured as follows. Chapter 2 presents both the high-level and detailed reuirements for a presence management solution in u-2010. Chapter 3 provides an overview of the developed Location Awareness System for the presence management requirements. The client side of the Location Awareness System is detailed in Chapter 4, with Chapter 5 concentrating on the server side. An evaluation and findings observed from tests of the Location Awareness System is presented in Chapter 6. Chapter 7 provides an alternative, or complementary method of providing presence management by using GSM Location Based Services and finally, Chapter 8 gives a verification that requirements identified in Chapter 2 were met.

1.1. Presence Management Definition

The term ‘presence management’ refers to the identification and monitoring of the presence or location of emergency workers and vehicles. The terms presence management and location management can be used interchangeably within the context of u-2010.

Generally, for a presence management solution we want to be able to accomplish two things:

1. Identify the emergency workers and vehicles that are responding to an emergency and monitor their progress towards the emergency location.
2. Monitor the movement of emergency workers and vehicles at the emergency location for the duration of the emergency operation.

One can also imagine that we may also wish to monitor the locations of emergency workers before an emergency occurs and determine their status. This could be especially useful in scenarios where only a skeleton crew of emergency workers are on duty at any given time or where the emergency workers are volunteers and respond to an emergency out of their work or leisure time. Thus, the status of these individuals can vary the nature of their vocation.

We will see that the presence management solution described in this deliverable is able to be used before, during and after an emergency operation according to the needs of the emergency organisation using it.

1.2. Advantages of Presence Management

A proficient presence management solution heralds several advantages in emergency and crisis situations. Furthermore, these advantages are amplified the greater the number of emergency workers, vehicles and distinct emergency organisations that are involved in the emergency operation. These advantages can include:

- Deploy emergency workers to specific locations more rapidly.
- React quickly to changes in the emergency context (e.g. redirect a group of emergency workers to a different location where another incident has occurred).
- Allow an emergency coordinator to quickly assess the status of any emergency operation by observing the locations of all workers and vehicles involved.

- Help an emergency organisation optimise the resources available resulting in more efficient emergency operations.
- Integrating the presence management solution with a communications network allows coordinators to quickly identify individuals they wish to communicate with based on their observed location (e.g. the nearest medic to a casualty or the nearest fire engine with cutting equipment to a car accident).

1.3. Presence Management in u-2010

Within the u-2010 project, the emergency scenarios to which a presence management solution is most applicable are the Fire in a Tunnel and the Mountain Rescue service. Both of these scenarios involve numerous rescue workers and vehicles that respond to emergency incidents. For the Fire in a Tunnel scenario, the presence management challenge is made a little easier since the destination for the emergency is known in advance and there is little movement around the location once emergency workers have arrived. In this scenario, presence management is primarily used to determine who is answering the emergency call and their progress towards the emergency location. Nevertheless, it is still advantageous for mission coordinators to know precise locations of workers and vehicles in and around the tunnel so that decisions can be taken in a more informed manner.

In contrast, the Mountain Rescue scenario has the most difficult requirements for which to design and implement a presence management solution. This is primarily because the problem domain is search and rescue. The location(s) where an emergency has occurred may or may not be known in advance and it is quite common for multiple rescue teams to be deployed to search a large area. Thus, we are presented with the situation that a mission coordinator must try to efficiently direct and manage the search patterns of numerous search groups to search a potentially large area of difficult and strenuous terrain.

Therefore, if we aim to solve the presence management requirements for the Mountain Rescue scenario, we will meet the Fire in the Tunnel requirements by default. One exception to this rule is that if the presence management solution relies on GPS, the Fire in the Tunnel scenario will require additional technology to overcome the lack of GPS signals inside the tunnel. This can be achieved with different location sensors placed inside the tunnel or by the use of GPS proxies inside the tunnel.

1.3.1. Presence Management in the Mountain Rescue Scenario

Figure 1 provides an overview of the network deployment model and how it relates to the general Mountain Rescue Scenario. Lancaster University is working in collaboration with the Cockermouth Mountain Rescue Team (CMRT) [8] to provide an on-mountain ICT solution with which the presence management solution will operate.

Each rescue worker that responds to emergency incident will carry a presence management device that will determine the location of the rescue worker and transmit the coordinates to the mission coordinator located at the headquarters (HQ) in the town of Cockermouth. However, in order for this to be possible there needs to be a communications network in place between the mountains and the HQ.

Lancaster University operates the MAN known as CLEO (Cumbria and Lancashire Education Online) which connects all the schools, colleges and universities in the counties of Cumbria and Lancashire to the UK academic network superJANET. The superJANET network is connected to GÉANT thus giving global IPv4/IPv6 connectivity. One of the objectives of the Mountain Rescue scenario in WP4 is to deploy strategic points of presence (PoP) that connect into the CLEO network and are in such a position that rescue vehicles have a good chance of connecting to them from the most frequent locations where they attend emergency incidents.

The Mountain Rescue HQ at Cockermouth is connected to the CLEO network, and so is turn connected to the global Internet via SuperJANET. The three CLEO PoPs are Lorton School, Gatesgarth Farm and Ennerdale Youth Hostel which are intended to be connected to CLEO in three consecutive phases (Lorton School is connected at the time of writing). From these PoPs the rescue vehicles can connect using 2.4Ghz wireless bridges. We have not discounted the possibility that some locations will not have line-of-sight to a PoP. In this situation, a temporary relay mast that can be rapidly deployed by a single person can be used to link the rescue vehicle and CLEO PoP. A rescue vehicle may also have a satellite dish and modem assembly to connect to the Internet via the Astra2Connect service. If this is used, the Astra 1E satellite will relay traffic to SES Astra at Luxembourg and then back through the global Internet to SuperJANET and CLEO.

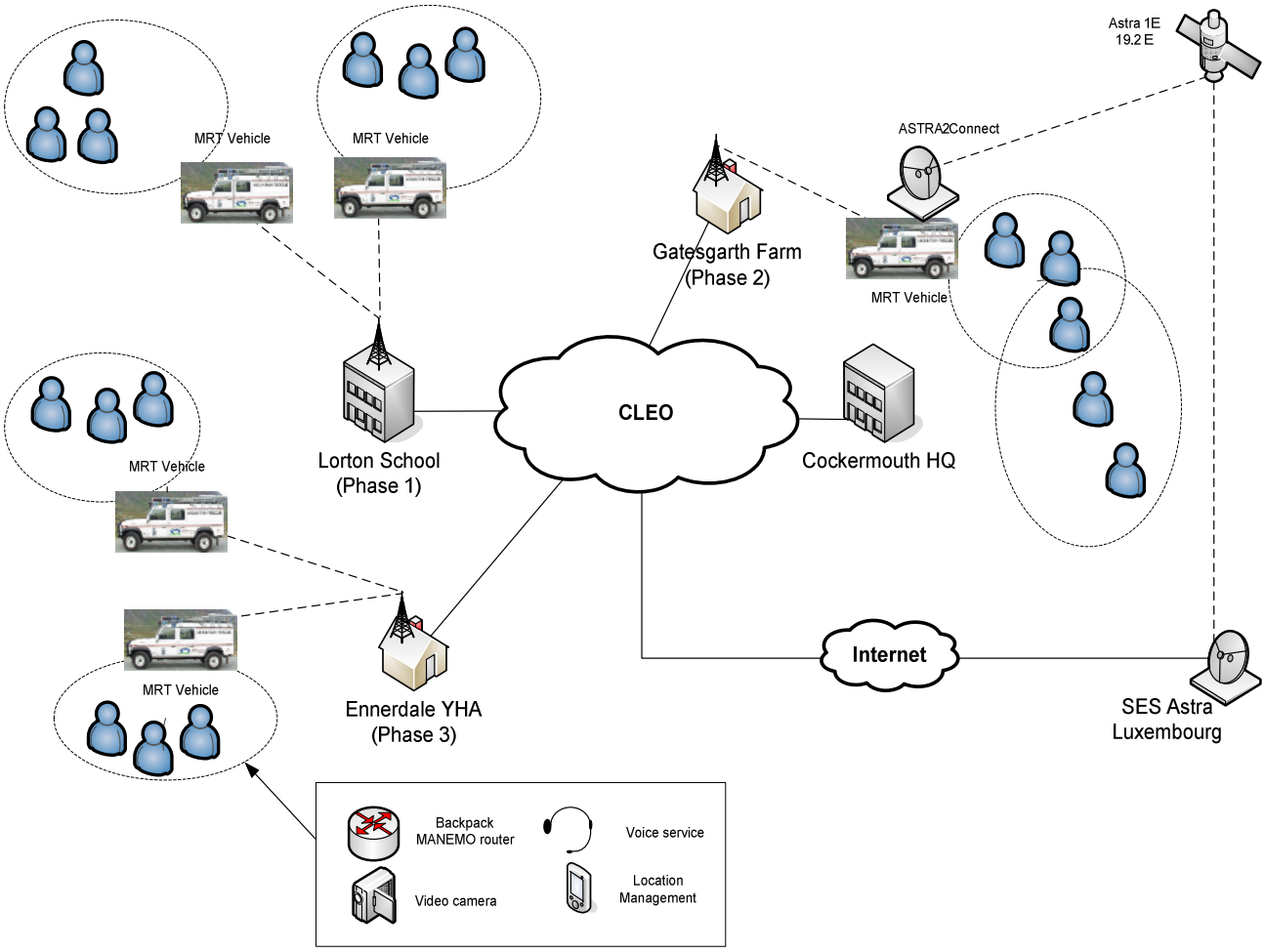

Figure 1 Network Deployment Overview

The rescue vehicles project their wireless hotspots in the search locations of their search groups. Each rescue worker has a backpack mobile router, voice headset and presence/location management device implemented on a PDA. Some rescue workers may also have helmet-mounted video cameras and biomed sensors for when the casualty is found.

For more details on the Mountain Rescue concept involving the network deployment and general ICT solution, please refer to u-2010 deliverable D4.2.1 [4].

2. Presence Management Requirements

The overall aim of the presence management solution for the Mountain Rescue scenario is to equip a mobile device with the capability of sending GPS coordinates to a server via the best-suited wireless communication that is available at that location.

Thus, the primary task of this piece of research is to develop an application for a suitable mobile device that the rescue workers will carry during their missions. This application will send GPS coordinates to the server that is located at the headquarters of the team.

Providing network services in a Mountain Rescue mission is a complex task primarily affected from the mobility of the end leaves of the mobile network (rescue workers), the mobility of the mobile network itself (search group) and the physical characteristics of the domain (mountains). Therefore, in order to achieve the described overall aim of the project, the following high-level requirements have to be achieved:

a) Identify, assess and prioritize connectivity options that could be used in a mountain rescue domain for transmitting GPS data.

b) Implement a client application for a mobile device that can recognise which of the defined connectivity options are available and efficiently utilise them for transmitting GPS coordinates according to some defined criteria.

c) Implement a server application that can receive the GPS coordinates that are sent from the clients, regardless of the communication option that a client used.

These high-level requirements regard three different aspects of this dissertation; identifying a framework regarding the communication options used, developing an application for the mobile device that the rescue worker will carry and finally, developing the application for the server located at the team's headquarters.

The first (a) high-level requirement described, demands a theoretical and practical study of the available connectivity options that can be used in the mountain domain that the Cockermouth Mountain Rescue Team operates in. This study should evaluate the connectivity options that could theoretically be used regarding the specific scope of sending the GPS coordinates, namely a fixed length string. This evaluation should also prioritise the viable connectivity options within the mountain rescue domain in order to provide a grounding for the application that will be developed on the mobile device. The underlying principle is to provide a basic connectivity framework with the options available and the rationale for using them, that could be used to aid the development of other networking services in such a domain.

The second (b) high-level requirement regards the client application that will be developed for the mobile device that the rescue worker will carry. This high-level requirement can be split into more detailed requirements concerning the client application's ability to:

1. Acquire real-time GPS coordinates.
2. Identify which of the predefined connectivity options are available.
3. Utilise the best-suited connectivity option and seamlessly swap among them based on some criteria.
4. Support both IPv4 and IPv6.
5. Transmit GPS coordinates in user-defined intervals.

Additional secondary requirements for the client application regarding the rescue worker's point of view would be that the chosen mobile device should be small and lightweight in order to be carried easily. Ideally, the mobile device should be wearable so that the rescue worker has his/her hands free. Moreover, the developed application for that device should be easily used without any configuration from the rescue worker.

Furthermore, the application should inform the rescue worker of its exact GPS coordinates, the wireless communication used at a time and the number of packets/messages sent using each communication method.

The third (c) high-level requirement described regards the application that will be developed for the server located at the headquarters of the team in Cockermouth. This high-level requirement can be split into the following more detailed requirements for the servers side application:

1. It should be able to listen for GPS coordinates regardless of the connectivity option that the client used to sent them
2. It should be able to identify the exact node that sent a message and the exact connectivity option that was used.
3. It should be able to identify messages received in the wrong order.

The main secondary requirement for the server application regarding the mission coordinator's point of view would be that the application should be able to inform him about the information retrieved. In particular, the server's application should present at least the GPS coordinates received overlaid on a suitable map, the ID of the node that sent them, the communication method that was used and the time that these were sent. Moreover, the server application should be able to count and present the content of messages received from different communication methods and present these data to the coordinator.

Keeping in mind the heterogeneity of the physical domain (mountains) and the unpredictable movement of the mountain rescue workers it becomes of great importance to develop a presence management solution that is flexible in terms of identifying, seamlessly swapping and utilising the available connectivity options according to the needs of the mountain rescue team.

The entire set of requirements for the presence management solution are summarised in Table 1.

Table 1 Summary of Requirements for the Presence Management Solution

Requirements for Presence Management	
R01	The client device must be able to acquire GPS coordinates in real-time.
R02	The client must identify which connectivity options are available.
R03	The client must utilise the best-suited connectivity option and seamlessly swap between the connectivity options based on pre-defined criteria.
R04	Both the server and client must support both IPv4 and IPv6
R05	The client must transmit GPS coordinates in user-defined intervals.
R06	The client device should be small and lightweight.
R07	The client device should be wearable.
R08	The client device should be waterproof.
R09	The client device should have a battery runtime of at least 4 hours.
R10	The server must be able to listen for GPS coordinates regardless of the connectivity option that the clients use to send them.

R11	The server must identify the exact node that sent a message and the exact connectivity option that was used.
R12	The server must identify and reorganise messages received in the wrong order.
R13	The server must display the locations of the client devices on suitable map displays.
R14	The server must log all movements made by the clients for later analysis.
R15	The client software should not need to be configured by rescue workers before use.
R16	The client GUI must be simple and utilise large buttons for use wearing gloves.

3. The Location Awareness System Overview

We have developed a Location Awareness System (LAS) that implements the presence management solution for the u-2010 Mountain Rescue scenario.

In the case of an emergency, the headquarters (HQ) of the team is notified, but often the exact location of the casualty is unknown. Therefore, the mountain rescue team is divided into independent search groups (composed of a number of rescue workers and all-terrain vehicles) that are distributed around the area in which the incident is suspected to have occurred. The mission coordinator of the team stays at the HQ and tries to remotely organise the rescue workers and improve the efficiency and the accuracy of the search and rescue mission.

Efficient collaboration and coordination of rescue workers in any search and rescue mission is essential, as every minute spent in the search can be critical. Our collaboration with the Cockermouth Mountain Rescue Team (CMRT) reveals that knowing the precise locations of search groups and their rescue workers during a mission is a significant aid to the mission coordinator and thus, a key requirement to increase the efficiency of the team.

The LAS we have developed informs the mission coordinator of the exact location of the members of the team in a real-time manner. Using the LAS, the mission coordinator can keep track of the rescue workers during the mission, and keep a broad picture of the search and rescue operation. This helps the coordinator take informed decisions in terms of which areas have, and have not, been thoroughly searched. Although the LAS has been specifically developed for the needs of the CMRT, it can be adapted to various search and rescue scenarios such as other mountain rescue domains or tsunami and earthquake aftermaths. The LAS is also applicable to other emergency and crisis scenarios such as the u-2010 Fire in a Tunnel scenario [6].

3.1. System Architecture Overview

The basic principle behind the LAS is the transmission of location coordinates, obtained via the Global Positioning System (GPS), from a client application that resides in small, lightweight devices carried by the mountain rescue workers. A server application, which runs at the team's HQ, receives these GPS coordinates and plots them onto a map, helping the mission coordinator to determine the status of the mission.

Figure 2 illustrates the high-level system architecture of the LAS. The system is composed of three conceptually distinct parts; the mountain rescue domain, which is the region that an incident can occur, the headquarters (located at Cockermouth, Cumbria, UK) and all the connectivity options that can interconnect these two domains.

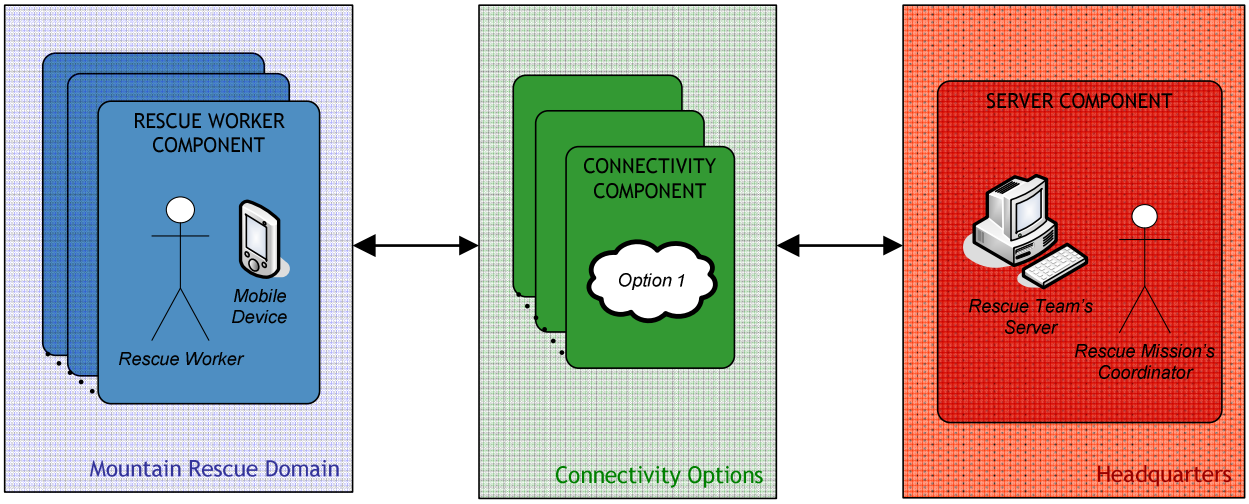

Figure 2 High Level system architecture of the LAS

As Figure 2 depicts, the mountain rescue domain includes many "rescue worker components", each one representing a rescue worker or vehicle with all the appropriate equipment. This equipment should be able to obtain GPS coordinates from dedicated satellites, then identify and utilize the best available connectivity option for the transmission of the coordinates to the server application. The implementation of the rescue worker component, i.e. the client application software and host device, is described in chapter 4.

Since a client device can conceivably communicate with its server peer in a variety of ways, it is necessary to construct a communication framework (namely, the different connectivity options) for passing location coordinates from client to server. This communication framework, including the format of location update messages, is described below in section 3.2.

Finally, the right-hand side of Figure 2 illustrates the HQ where the mission coordinator monitors the server application, which is responsible for receiving, sorting and mapping location coordinates. The implementation of the server component is described in chapter 5.

3.2. Communication Framework

In general, the mountain rescue missions take place in areas that are sparsely inhabited and consequently, have an acute lack of fixed network infrastructure. There are no WiFi or WiMAX hotspots, GSM/GPRS coverage is sporadic and 3G coverage is non-existent. The sporadic nature of GSM/GPRS coverage is further damaged by poor signal propagation due to the geographic terrain (multipath propagation and non-LoS to the base transmitters due to the presence of hills, mountains and trees).

However, one of the primary goals for the u-2010 mountain rescue concept [4] is for the rescue team to be able to rapidly construct their own wireless network in an ad-hoc fashion to cover their search area. Thus, the client devices carried by mountain rescue workers must have the ability to use the rescue team's wireless network in addition to any fixed coverage that is available.

However, even the rescue team's own network will exhibit unreliable characteristics as rescue workers move further away from the transmitting base and lose LoS with each other due to distance and/or geographic features. Thus, the client devices must also be able to react to network conditions and select the must suitable network that is available at any point in time.

For these reasons, the LAS, as shown in Figure 3, includes several connectivity components that can be used for the transmission of the GPS coordinates from the client applications to the server.

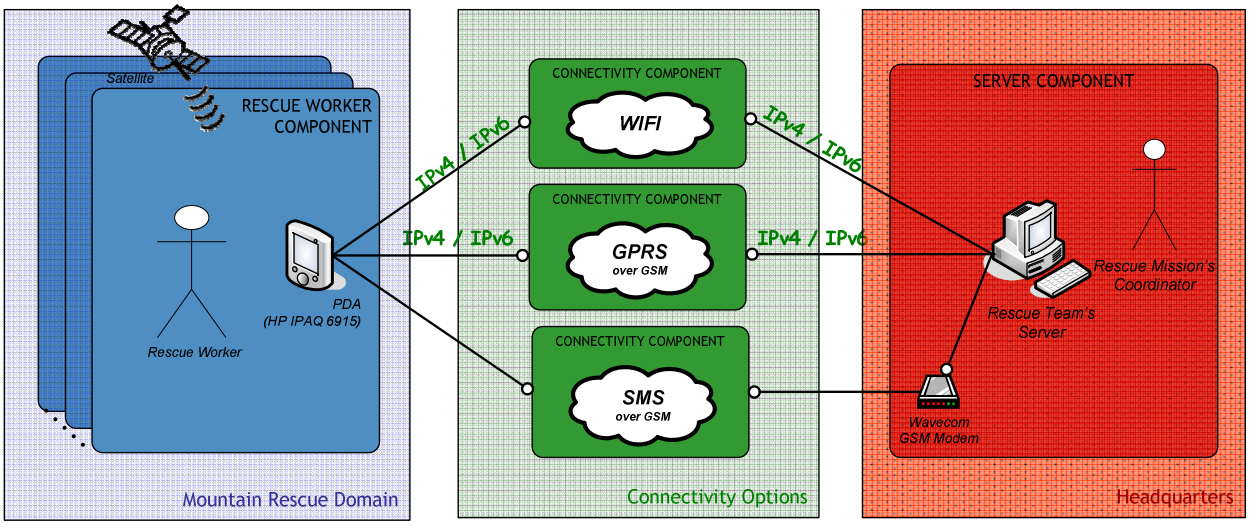

Figure 3 Detailed System Architecture of the LAS

3.2.1. WiFi

The Network Mobility group at Lancaster University [9] has developed a network communication model that includes a mobile WiFi based network infrastructure to provide coverage in the areas that the search parties roam. This WiFi coverage is provided primarily by 802.11 wireless hotspots that are projected from high-gain antennae located with the all-terrain rescue vehicles parked close to the incident. This coverage is extended from the mobile routers that a member of each search party carries to provide a shorter-range wireless hotspot to facilitate communication between members of the search party.

The WiFi connectivity option and the way it is used presents some significant advantages over other communication methods. For example, it can be considered highly flexible, as the wireless hotspots projected from the rescue vehicles can use directed antennae to focus coverage in the intended search area, rather than in an omnidirectional fashion. Coupled with the use of backpack mobile routers, this allows connectivity to be available areas where there is very poor or non-existent GSM/GPRS coverage.

In addition to this, the WiFi based mobile network equipment is owned by the members of the team and thus it can be configured and managed according to the exact needs of the team. For example, the routers of this team can be configured to take advantage of IPv6 and the NEMO basic support protocol. This allows the team to bypass the normal delays associated with adoption of new communication protocols by third-party telecommunication companies. Furthermore, the WiFi connectivity can be considered less expensive than other options because the equipment is not leased from any third-party companies and it does not include any billing upon usage. Thus, the primary cost for WiFi is during equipment purchasing, as the operating costs are minimal.

However, the WiFi connectivity option cannot guarantee full coverage for every member of a search team due to the geographic nature of the search terrain. Team members will have to move away from each other to cover large search areas and even the extensibility of the mobile router approach may not provide adequate coverage for the entire search area. The relatively high frequency and low EIRP of WiFi transmissions mean that coverage is easily lost due to large distances and obstacles blocking LoS between stations.

Consequently, we have defined two additional connectivity options that can provide redundancy for the transmission of the GPS coordinates and improve the effectiveness of the overall communication model.

3.2.2. GPRS

One of the alternative options that has been defined to complement the primary WiFi connectivity option is to use the General Packet Radio Service (GPRS). GPRS is a mobile data service that can be used to transmit and receive data over the GSM network. Usually, GPRS is used by individuals to have access to WAP, MMS or the Internet from mobile phones and PDAs. This is normally in a packet-switched fashion, meaning that they are charged only by the amount of data they transmit/receive. Older GPRS implementations, which are still provided by telecommunication providers, are using the circuit switched data (CSD) standard billing the user per minute independent of whether he/she is actually transferring data.

GPRS is a powerful connectivity option because it can be used wherever there is sufficient GSM coverage. Major UK telecommunication providers claim that they have networks covering over 99% of the population. However, considering the region that we are interested in is very sparsely populated and that the presented percentage is usually a by-product of marketing policy, this statistic can be misleading.

In reality, much of the region falls into the category of "variable quality service" when viewed on coverage maps provided by the mobile phone companies. From experience using GSM/GPRS equipment in the area, this can vary from no signal at all up to signals good enough for standard GSM voice calls and lower class[1] GPRS connections. The areas where there is no GSM signal are in the minority and are usually in low-lying valley areas.

A significant advantage that GPRS presents is that it is an IP-based solution that, in theory, can track the evolution of IP-based network protocols. Therefore, although current UK GSM telecommunication companies provide only IPv4 support over their GPRS network, we anticipate that they will migrate to more innovative solutions like IPv6. This movement would inherit the benefits of Mobile IPv6 and provide greater support for the network mobility concept of our domain. Moreover, the IP-based nature of GPRS allows telecommunications providers to upgrade their data services (e.g. from GPRS to EDGE or UMTS) whilst still providing the same interface to applications. Thus, an upgrade of data service to provide higher data rates will not require any change to the design of our system.

A critical aspect of this connectivity option is the potential data rate that it can provide. GPRS was developed to provide data rates from 10 to 100Kbps based on the class of the GPRS client hardware and the quality of the GSM signal. It should be mentioned that GPRS packets are prioritised below voice calls and consequently, encounter relatively high latency and round trip times compared to standard Internet connections. In addition, the data rates offered are not always stable as they are dynamically affected by the GSM signal quality. However, for the focus of transferring GPS coordinates, even the lowest class GPRS connection will suffice, as the payload being transferred by the LAS client software is a string of less than 100 characters (discussed further in section 3.2.5). For this reason, the proposed GPRS connectivity option is also considered to be relatively financially affordable, as the CMRT will be billed only for the amount of data transferred.

3.2.3. SMS

The well-known and extremely popular Short Message Service (SMS) is presented as the third connectivity option that can complement the envisaged scenario in a mountain rescue domain. The data that the client application transfers can be represented in a defined fixed-length character format that can be sent as an SMS message from a mobile device.

[1] E.g., class 1, 2 or 4.

The use of the SMS connectivity option depends on the availability of the GSM network. Although GPRS also shares this characteristic, the two connectivity options should be viewed as separate. This is because GPRS services are not always available even if there is a sufficient GSM signal quality; this is especially true in some rural areas. In addition, an existing GPRS connection may be dropped due to degrading GSM signal quality or by it being pre-empted by voice calls. Yet, in such circumstances, it is entirely possible that SMS messages can be still sent and received without any problems; an SMS message can often be sent even when the GSM signal strength is very poor. Therefore, it is clearly acceptable for SMS to be used as redundant connectivity option when GPRS is not available or the connection is dropped.

Moreover, a GPRS connection needs at best 10 to 15 seconds to be established even with excellent GSM signal strength. Conversely, a SMS message can be sent from a mobile device as soon as the message is composed. In addition, SMS is a method that does not require any set up fees and can be relatively cheap if used as a backup connectivity option.

3.2.4. Prioritizing The Connectivity Options

The central part of Figure 3 depicts the three most suitable communication options in our mountain rescue domain to transmit GPS coordinates back to the HQ. The first option is to transmit the coordinates via IPv4 or IPv6 over an 802.11 WiFi network, the second option is via IPv4 or IPv6 using GPRS over the GSM network and the third option is to transmit them via SMS over the GSM network. The client application needs to identify the availability of each connectivity option and use them according to this priority i.e.:

1. WiFi
2. GPRS
3. SMS.

This priority has been determined by comparing and contrasting the different connectivity options based on the following criteria:

- Availability
- Cost to purchase/install
- Operating cost
- Extensibility of the provided service
- Ability to follow evolution.

The WiFi connectivity option receives the highest priority since, in our mountain rescue domain, it is the most likely to be available as it is provided from the equipment that the rescue team carries. The fact that the coverage follows the mobility of the rescue team also means that the WiFi network is the most likely have suitable signal coverage. In addition, WiFi is a connectivity option that, although has a relatively high purchase cost, has minimal operating costs. It also provides service extensibility since it is IP-based. Consequently, it has the ability to follow evolution, for example use MANET, Mobile IPv6, NEMO and MANEMO protocols.

Complementing WiFi with the GPRS option seems promising as GPRS relies on a different network infrastructure, the GSM network. Like WiFi, GPRS is an IP-based solution that, in theory, could also follow the evolution of IP-based networks, providing extensibility for the service offered.

The SMS option receives the lowest priority of the three, essentially because it is not an IP-based service and has a very limited character set. However, we can still employ it as the defined message format (described in section 3.2.5) for transferring GPS coordinates is a fixed set of simple characters.

The various pros and cons of the different connectivity options are summarised in Table 2.

Table 2 Summary of Connectivity Options

	WiFi	GPRS	SMS
Availability	Good	Poor	Moderate/Poor
Purchase Cost	High	Low	Low
Operating Cost	Low	Moderate	Moderate
Service Extensibility	High	Moderate	N/A
Ability to follow evolution	High	Moderate	N/A

WiFi availability is considered to be good by the fact that it is provided by special equipment that "follows" the rescue workers' mobility. However, WiFi availability is dependant on how well the communication model is applied to the Mountain Rescue domain and if the mobile equipment is in place at suitable locations. Outside the targeted search domain, rescue workers will have no connectivity to the WiFi network. In contrast, GPRS and SMS have blanket coverage of the area but the quality of the coverage can vary from non-existent to perfectly adequate. SMS will generally fare better than GPRS as SMS messages can be sent even with very poor GSM signal quality.

For purchase costs, the WiFi coverage is provided by mobile routers and directed antennas, equipment that is generally expensive to purchase but cheap to operate. On the other hand, GPRS can have low or zero cost to set up while its running cost is dependant upon usage. Similarly, the SMS connectivity option does not need any setting up fees and its running cost is also usage dependant. However, taking common current-day tariffs into account, SMS will cost more per-byte than GPRS when used regularly to send data. The running cost of the SMS service can be relatively high if it is not used with caution[1].

The WiFi and GPRS connectivity options provide, in theory, great extensibility for the type of the service that they can support. Although the defined data that is transferred is a fixed set of characters, as both methods are IP-based they can truly support any type of network service that can be facilitated from the IP protocol. For example, both communication options can generally support the transmission of photos or complex data types if such a service is needed for the team in the future. Thus, the LAS can be extended to provide richer and complex content should the need arise. However, The SMS connectivity option cannot provide any extension as the only data type/service that it can provide is transferring a limited set of characters.

The WiFi connectivity option is able to follow the evolution of networking state of the art. For example, the WiFi based network can use IPv6 and can have all the advantages that Mobile IPv6, MANET and NEMO basic support can provide. Furthermore, as the WiFi network is provided with equipment owned by the Mountain Rescue Team it can be configured and tweaked to their exact requirements. GPRS also has the ability to follow evolution but this is not as feasible as WiFi since it is essentially a point-to-point protocol and suffers from traditional slow adoption of new protocols by incumbent telecommunications providers. Of course, SMS is not able to follow evolution at all.

In summary, the WiFi option is considered to be the primary option as it is the most tailored towards the mountain rescue team's requirements, provides excellent extensibility and has the cheapest operating costs. The GPRS connectivity option is considered to be the next preferred option as it can potentially

[1] It has to be said that the cost criteria is relatively subjective. On one hand, the cost of say 500 SMS messages may seem unimportant in relation the successful recovery of a casualty. On the other hand, most Mountain Rescue teams are funded by voluntary contributions. Thus, operating costs are a prime concern.

provide greater networking services than the SMS option and also being less expensive to operate than SMS. SMS is therefore considered mainly to provide redundancy for the two IP-based options when they are unavailable.

3.2.5. The Unified Message Format

An important feature of the communication framework is that it defines a Unified Message Format (UMF) that is used for the transmission of the GPS coordinates from the clients to the server regardless of the connectivity option used. This simplifies the procedures for the transmission, reception and processing of the packets/messages both on the client and the server side.

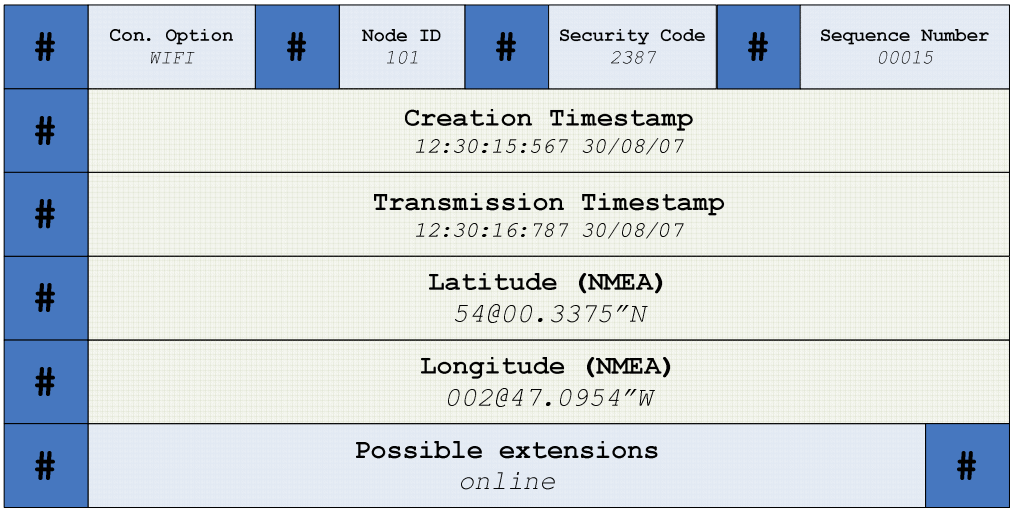

Figure 4 Unified Message Format

Figure 4 illustrates the UMF that is used for the transmission of GPS coordinates. As depicted, the defined format of each message is composed of nine different fields separated with the character "#". The use of a special character to separate the fields instead of defining a fixed length for each field was preferred for two basic reasons. Firstly, some fields, such as "Connectivity Option", "Timestamp" and "Possible extensions" do not have a fixed length and the use of the special character eases their separation on the server side. Secondly, all the messages being sent and received are logged on both sides and the use of a special character eases their readability, instead of having the fields packed together.

The defined format of each message consists of the following fields:

1. Connectivity Option: This field describes the connectivity option that is used to send the message. Nominal values for this field are WiFi, GPRS and SMS.
2. Node ID: This field includes the Node ID of the client that sent a message. The Node ID is a three-digit number that uses the first (from the left) digit to describe the ID of the search party of the client with ID described by the two remaining digits, belongs to.
3. Security Code: This field includes a four-digit number that is unique to every node and is used as a security code to prove to the server the authenticity of the node that sent the message. This security code can be changed from the GUI that each client has.
4. Sequence Number: This five-digit field describes the sequence number of the message being sent from one client. The server can use this number in conjunction with the node ID of a message to identify messages received in wrong order.

5. Creation Timestamp: This field describes the time and the date that a GPS location was logged by the device. The time value follows the format “HH:mm:ss:msec” and the date the format “dd/MM/yy”. These values are acquired from the Operating System of the PDA. Creation timestamps are used primarily for the correct sequential display of movement patterns at the server.
6. Transmission Timestamp: This field describes the time and the date that a message was sent from the client. The time value follows the format “HH:mm:ss:msec” and the date the format “dd/MM/yy”. These values are acquired from the Operating System of the PDA. Note that the difference between the creation and transmission timestamps can be used to detect problems with the connectivity option being used.
7. Latitude: This field includes the latitude part of the GPS coordinates that the client sends in the NMEA format. An interesting point is that the latitude in NMEA format should be of the type, (for example) “54°00.3137"N”, but the character "°" cannot be sent in an SMS message. A simple workaround was developed to replace, on the client’s side, the character “°” with “@” to all the transferred messages and modified back to “°” on the server side to keep a uniform format.
8. Longitude: This field includes the longitude part of the GPS coordinates that the client sends in the NMEA format which should be of the type, for example, “002°47.3053”. Again, the character "@" was used in place of character "°" to preserve a uniform format.
9. Possible extensions: This field can include any additional information that the client decides to send and is there for future use. The current use of the field identifies if a message is transmitted online or offline (these are explained in chapter 4).

The defined UMF, apart from simplifying the process of transmitting and receiving coordinates, includes valuable data, which can enable significant features to take place. For example, the server application can do a simple authentication mechanism for the client that sent a message, based on the values of the node ID and the security code fields. Moreover, the server application can re-order out-of-sequence messages based on the node ID and the sequence number of each message. In addition, an identification mechanism can take place as the node ID field of each message maps to a certain device, which in turn and with the aid of our database infrastructure or directory services, maps to a certain person or vehicle. Therefore, the server application can identify each rescue worker and his/her position and overlay this information onto suitable maps.

4. LAS Client Implementation

The client side of the LAS is composed of lightweight, small end devices that are carried by the rescue workers during a mission. The aim of the application running on these devices is to obtain GPS coordinates, identify the best-suited connectivity option and use that option for the transmission of GPS coordinates to the server application. Consequently, the client devices should be GPS, WiFi, GPRS and GSM enabled to be able to utilize the connectivity options. For the purpose of our research, we are currently using HP IPAQ 6915 PDA devices running Windows Mobile 5.0 that, in principle, can support the requirements of our devised system.

Each of the client devices run an application that is implemented in Visual C# using the .NET Compact Framework, libraries from the OPENNETCF SDK and calls to native DLL files of Windows Mobile 5.0. Generally, the application running on the client is a multi-threaded application for the efficient implementation of concurrent activities. Its functionality is based on timers (e.g. sending coordinates at specific intervals) and events (e.g. the acquisition of GPS coordinates).

The client application includes five different tab-pages:

1. Main - the main screen of the application.
2. Client - for settings regarding the client.
3. Server - for settings regarding the server.
4. Map - for displaying a map to the rescue worker showing his/her location.
5. Info - information obtained from the GPS satellites.

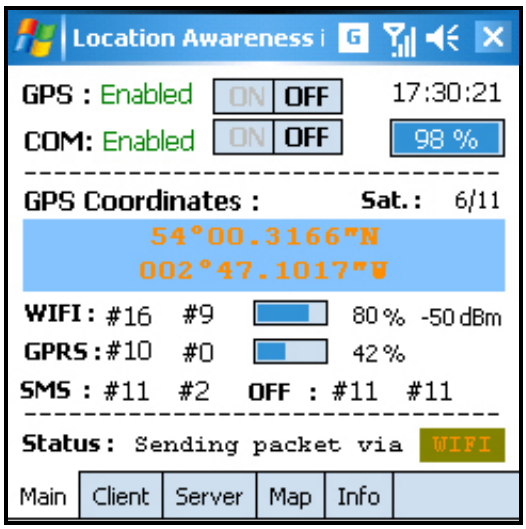

Figure 5 Main Tab of Client Application

The Main tab, depicted in Figure 5, is the main screen that the mountain rescue worker sees during a mission. This screen presents information for:

- his/her location
- the number of GPS satellites that the application can lock and view
- indications for the WiFi and GSM signal strength

- the current connectivity option being used
- counters for the messages being sent from each connectivity option or stored for later transmission.

The application can also display the walking speed of the rescue worker holding a device or the driving speed of a vehicle that the rescue worker is in, based on data retrieved from the GPS module.

The client application is able to work seamlessly in two basic modes: online mode and offline mode. Online mode includes the transmission of the coordinates when at least one connectivity option is available. In contrast, when no connectivity option is available, the application works in offline mode and stores messages for later transmission. The transition from one mode to another is done automatically, without any intervention from the rescue worker and does not restrict the two modes from running simultaneously. The decision regarding whether a connectivity option is available is based on a complex mechanism that takes into account the signal strength of each connectivity option, the ability to perform a connection (for the IP-based connectivity options) and the monitoring of the actual transmission.

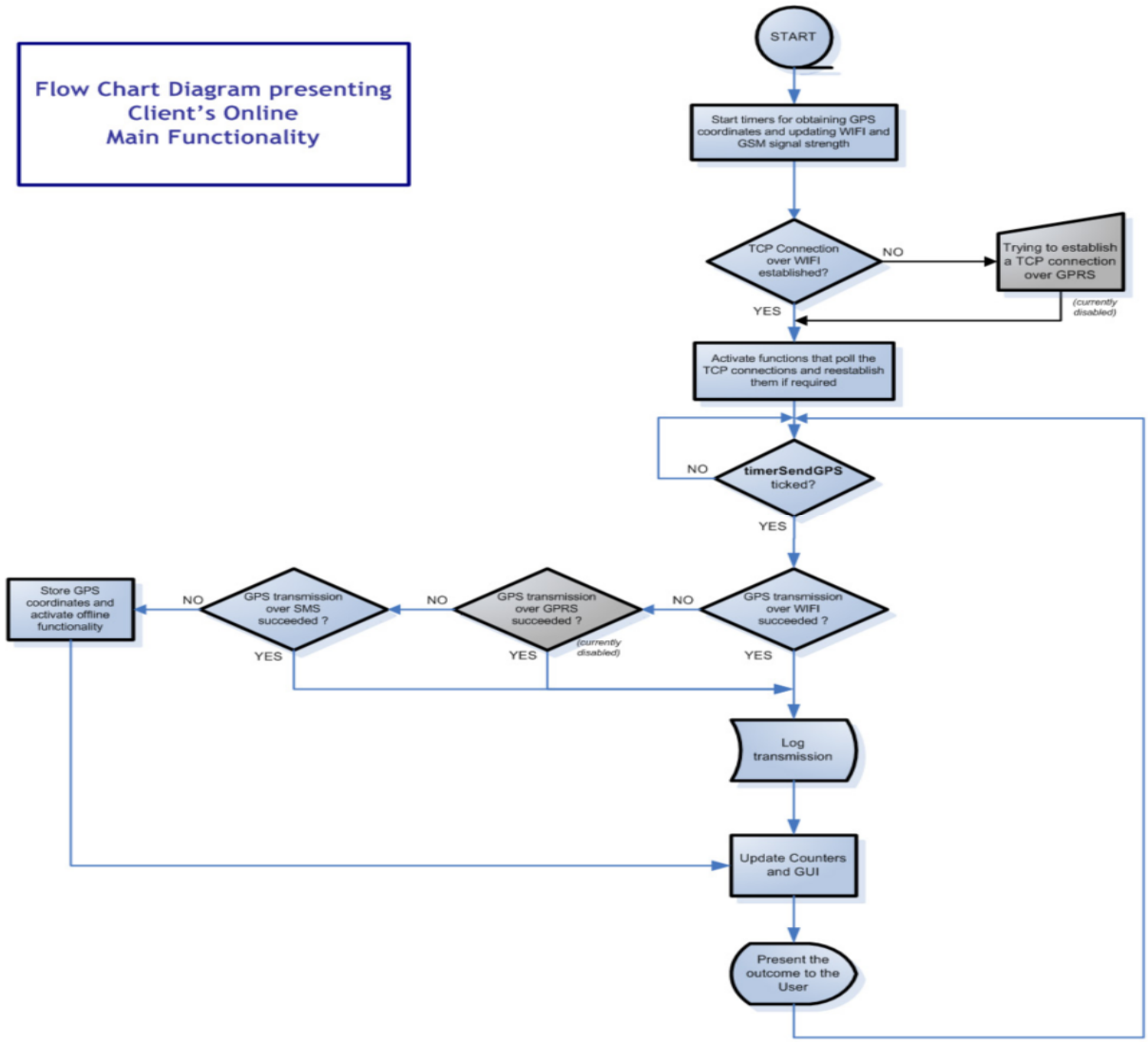

Figure 6 Online Mode Flow Chart

The client application has an efficient mechanism to constantly monitor and poll the IPv4 and/or IPv6 TCP connections that are used, to keep them active and re-establish them in the background if required. When none of the IP-based options are available then the SMS connectivity option is used as a backup. The transmission of SMS messages is also monitored for success. If the SMS transmission fails, the application enters offline mode. In offline mode, the application monitors the availability of the different connectivity options and, when connectivity is regained with at least one of them, sends the stored messages flagged as offline. This is important since offline messages represent locations that a rescue worker has been in the past. At this point, it should also be mentioned that the online functionality continues working in parallel with offline as any live GPS updates must still be sent to the server application. Figure 6 and Figure 7 show the logic of the online and offline modes respectively.

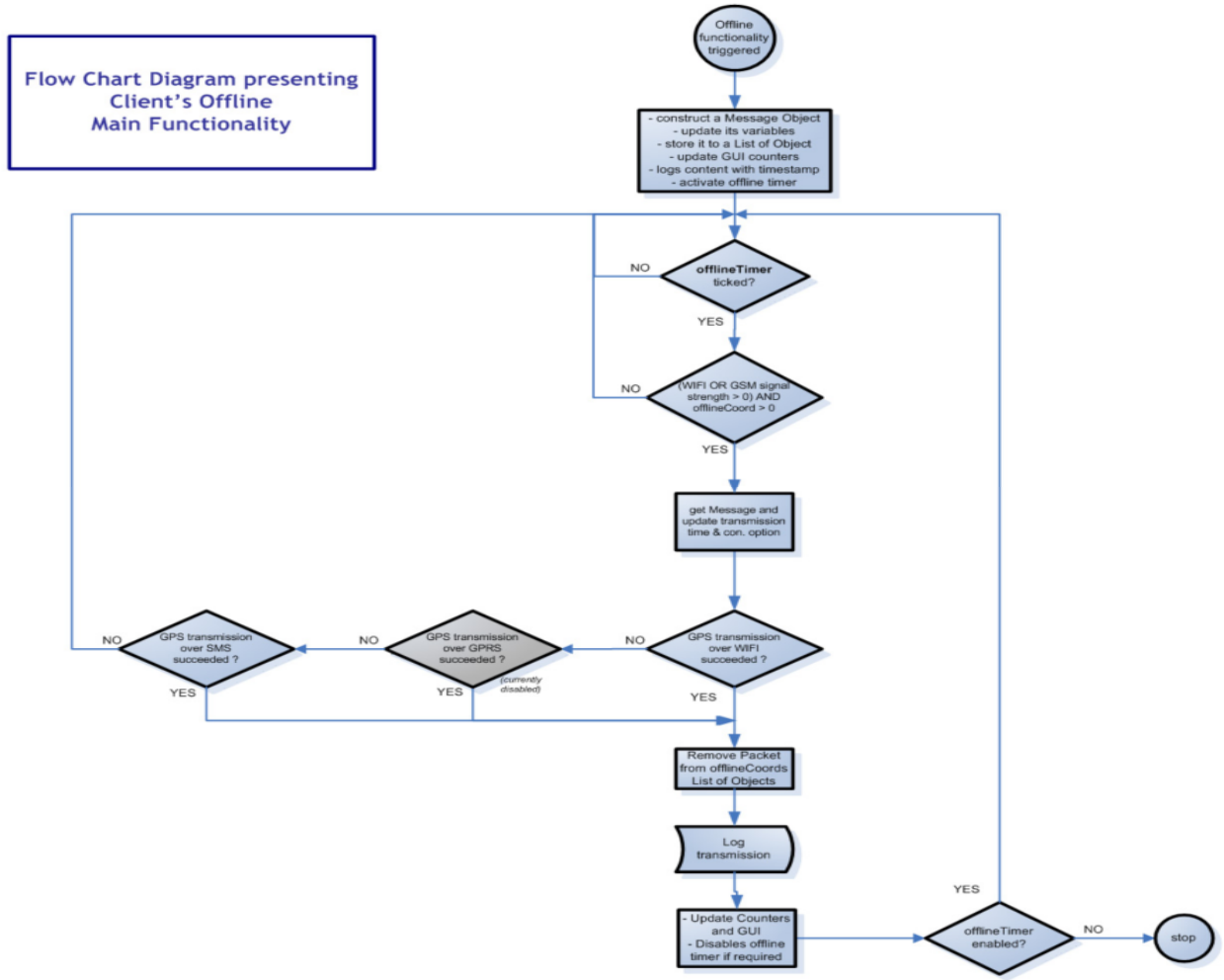

Figure 7 Offline Mode Flowchart

The client application supports on-the-fly modification of the transmission interval used for sending the GPS coordinates. This is useful for reflecting the various stages of a search and rescue operation. For example, rescue workers in vehicles en-route to a search location will travel much faster than walking pace and consequently, more frequent updates are needed to monitor their progress. Similarly, a client device attached to a search and rescue dog will need to send updates more frequently than that of a rescue

worker at walking pace. The application also includes comprehensive and frequent logging of all valuable data observed on the client device. In combination with the server application, this enables future replaying of search and rescue missions for analysis by the mountain rescue team.

Furthermore, the availability of each connectivity option, in conjunction with GPS coordinates, is logged to give us the ability to build coverage maps and improve the coverage of the connectivity options if and when this is possible. This information can also be fed to our search theory system (see [4]) to increase the efficiency of future missions and arm the LAS with the functionality to underline regions with higher possibilities of finding casualties.

Figure 8 presents a high-level class diagram and their abstract interconnection for the client application. The `Program` static class is responsible for launching the application and creates an object of the `GUI` class, which provides the main functionality. The `GUI` class includes a GUI form to hold all the elements that are presented to the rescue worker and also instantiates one object of the `GPSConnector` class and one of the `IPConnector` class. The `GPSConnector` class provides the connection with the GPS device and instantiates an object of the `NMEAInterpreter` class. The `GPSConnector` initiates a connection with either the external GPS device via Bluetooth or the internal GPS device of the iPAQ 6915 and continuously retrieves data using serial communication. This data is passed to the `NMEAInterpreter` object, which launches events upon the information retrieved.

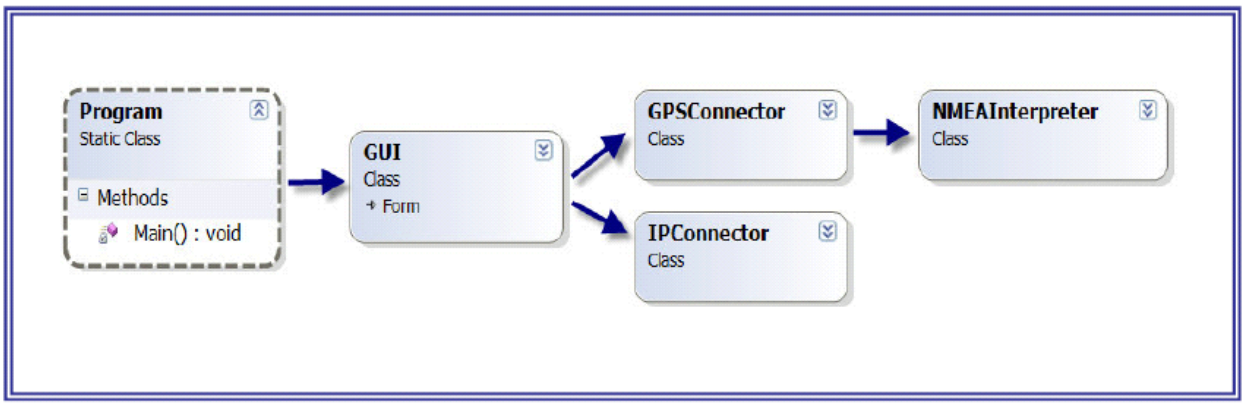

Figure 8 High Level Class Diagram for the Client application

The remainder of this chapter describes the main client components. In particular, section 4.1 describes the `GUI` class, section 4.2 the `GPSConnector` class, section 4.3 the `IPConnector` class, section 4.4 the Map feature, section 4.5 the Log feature, sections 4.6 and 4.7 describe the hardware and software choices respectively and finally, section 4.8 describes the main restrictions that the client hardware and software presented.

4.1. GUI

The GUI object of the GUI class is instantiated from the main method of the `Program` class and holds all the elements that the user can see, along with the main functionality of the client application. This large class, composed of about 3500 lines of code, is split into two distinct logical and practical parts taking advantage of the partial class implementation feature of Visual C#. The first part of the GUI class is the `GUI.Designer.cs` (automatically named by the Visual Studio) including all the GUI elements that the rescue worker can see and interact.

The second part is the partial implementation of the GUI class including the main functionality of the application developed for the client.

At this point it should be mentioned that this separation was followed in order to keep the functionality of the program as effective and simple as possible, since the `GUI.Designer` and the partial `GUI` main functionality implementation are constantly passing arguments to each other. An alternative solution could be to create a distinct class for the main functionality of the program but this would impose the implementation of many more background threads and methods to pass arguments from one thread to another; a process that would consume much more of the limited resources of the PDA.

4.1.1. GUI Designer

The GUI Designer includes many variables and methods to represent, update and handle the elements of the GUI Form that is presented to the rescue worker. Moreover, these functions handle the interaction of the user with the GUI and the events that are triggered when GPS data is received.

The GUI of the client's application is mainly composed of five different tab pages to separate distinct parts of the application, namely Main tab, Client tab, Server tab, Map tab and Info tab.

Figure 5 presents the Main tab of the application, which is the main screen that a rescue worker will see during a mission. In the upper left corner of the Main tab, two basic functionalities are presented to the user, GPS and communication (COM), which can both be turned on and off using the appropriate buttons. If the user turns the GPS functionality on then a connection is established with either an external GPS device or the internal GPS module that the PDA has, based on the options chosen on the Client tab (described below). If the user turns on the communication functionality (which can be done only if the GPS functionality is on), then the client will start transmitting GPS coordinates using the best-suited connectivity option at every transmission interval (default is 15 seconds). The upper right corner of the Main tab presents the current time and the capacity of the battery of the PDA.

The central part of the Main tab presents the most recent longitude and the latitude of the GPS coordinates received from the GPS module in NMEA format. These values are updated every time an appropriate signal is received from the GPS module and transmitted to the application based on the anonymous delegates, an inherent functionality of Visual C# that enables methods to be called when events are fired. On the right side of the central part of the Main tab the satellites that are used (locked) from the ones that can be viewed are presented.

Below the GPS coordinates, counters for the messages that are sent using each connectivity option are illustrated to the user. Adjacent to the WiFi messages sent counter, the WiFi signal strength is presented in both graphical and numerical format (db).

At the bottom of the screen, above the tab options, an informative message is given according to the functionality of the application at any given time. For example, messages like "`Sending packet via (…)`", "`Changing networks`", "`Found eduroam`" or "`Trying to reconnect`" are likely to be presented there. Finally, at the bottom right corner of the Main Tab the current connectivity option used is presented, namely WiFi, GPRS or SMS.

The left-hand side of Figure 9 depicts the Client tab page, illustrating the settings for the client's side of the application. On this tab, the rescue worker can change the Node ID of the device and its Security Code. On this tab there is an option defining if the application will receive GPS coordinates from an external Bluetooth device or the internal GPS module of the iPAQ device.

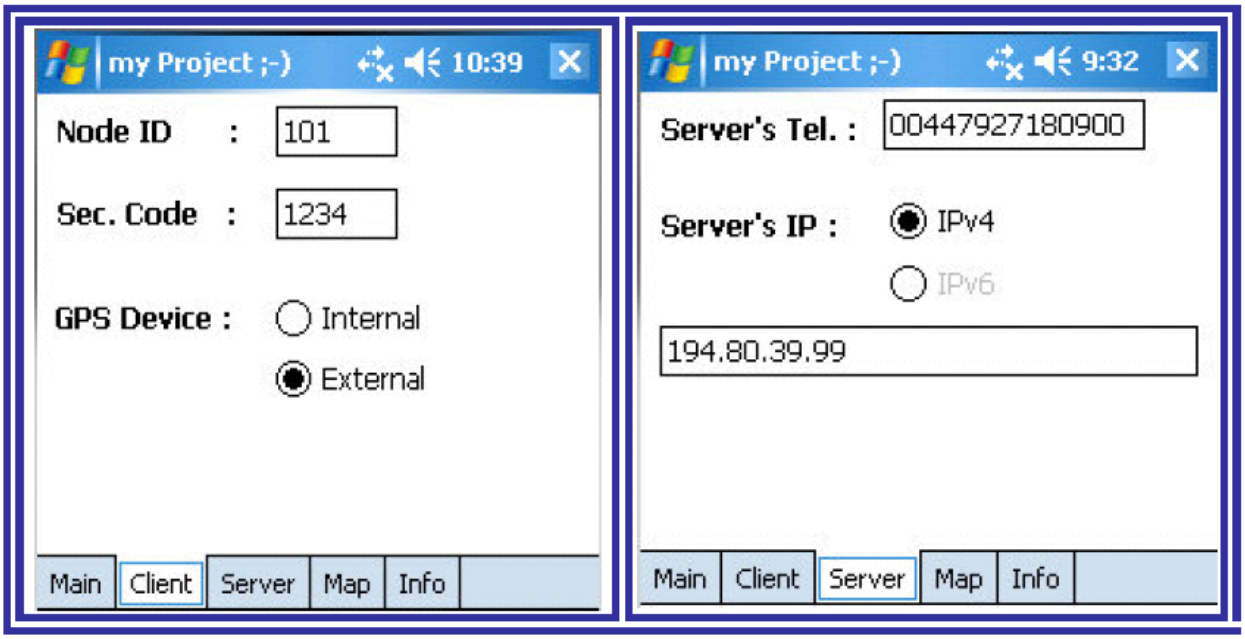

Figure 9 Client tab page (left) and Server tab page (right)

The right-hand side of Figure 9 depicts the Server tab page, which includes settings regarding the server. In particular, it contains the telephone number to which SMS messages should be sent and the IPv4/IPv6 addresses of the server.

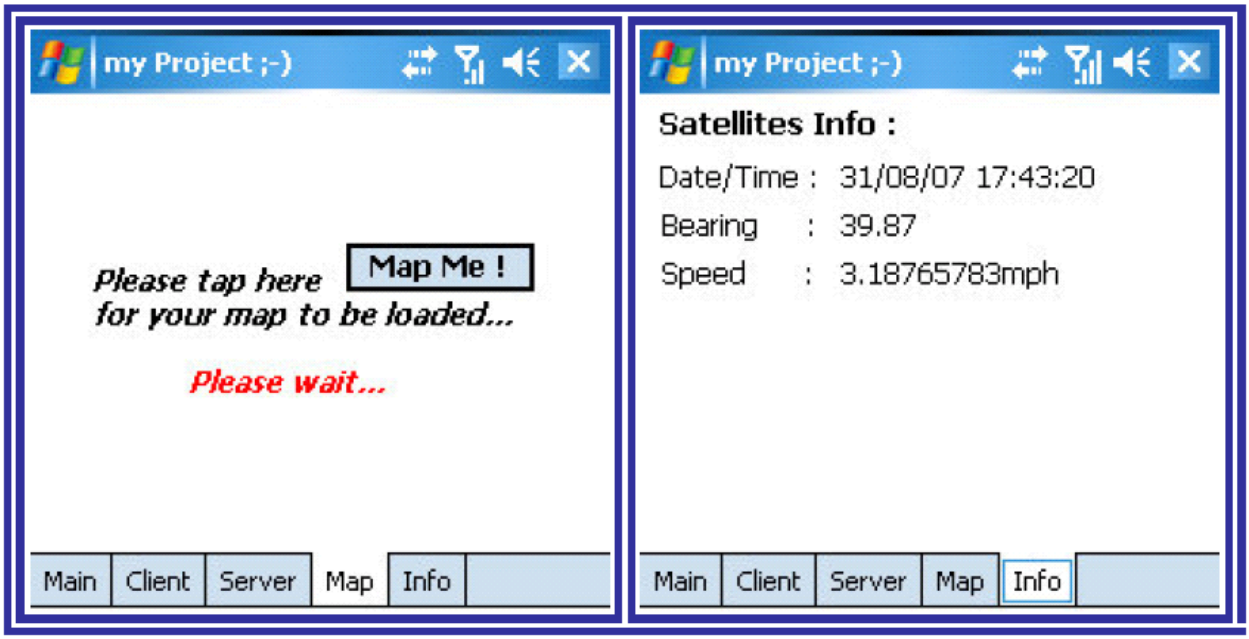

Figure 10 Map tab page (left) and Info tab page (right)

The left-hand side of Figure 10 shows the Map tab page that gives the rescue worker the ability to map his/her precise location based on the most recent GPS coordinates displayed on the Main tab page. The

right-hand side of Figure 10 illustrates the Info tab page, which presents the rescue worker with information derived from the GPS module, such as the current date, time, bearing and the current speed of the person holding the PDA.

4.1.2. GUI Main Functionality

This section describes the main functionality of the application, which is the partial implementation of the GUI class of the client. Upon the creation of the object of the GUI class, the following five actions are taken:

1. Initialisation of all the `GUI` elements.
2. Initialisation of all the GUIDs that are the references that the class will hold to refer to either the WiFi or the GPRS connection.
3. Initialisation of the log file.
4. Creation of a `GPSConnector` object that will represent the connection with the GPS module.
5. Creation of an `IPConnector` object that will represent the most suitable IP connectivity option at any time.

When all the above actions are performed successfully, the application waits for the rescue worker to enable the GPS functionality by tapping the appropriate ON buttons (Figure 5). If the user enables GPS and has not modified the default option to use an external GPS device, he will notice a screen with all the available Bluetooth devices that the PDA can see. If the user has chosen the internal GPS module from the Client tab (Figure 9) then he should also wait for the PDA to initialize the internal GPS module. From this point on, the application will display to the user GPS information that will be received from the GPS module.

Consequently, the rescue worker will tap the ON button to start the communication phase and instruct the device to transmit GPS coordinates to the server. At this point, the application performs various checks to identify if the WiFi connectivity option is available. If the WiFi connectivity option cannot be used at that time, then the device will try to initiate a GPRS connection. If either of the previous IP-based connectivity options is successful then the device will try to open a socket with the server over the successful connectivity option and inform the user that he has been connected to the server. Whether the previous procedure has been successful or not, the application will initialize two different timers, which will be used for future communication.

The first timer, named `timerWifiSignalStrength`, ticks every two seconds and is responsible for updating the WiFi signal strength indication on the Main tab page. In order to achieve this update it performs the following actions in order:

1. Acquires a reference of all the network interfaces
2. Identifies the wireless network interface
3. Checks if the PDA has been connected to the defined WiFi network
4. Checks if the PDA has obtained a valid IP address
5. Acquires the signal strength (in db) of the wireless adapters and updates the WiFi signal indication on the screen.

The acquisition of the WiFi signal strength is implemented with the use of the OPENNETCF library [11] based on the indications that the WiFi network driver can provide to the developed application. This WiFi signal strength indication can obtain one of the following distinct levels:

- excellent signal (higher than -50db)
- very good signal (lower than -50db and higher than -70db)
- good signal (lower than -70db and higher than -80db)
- poor signal (lower than -80db).

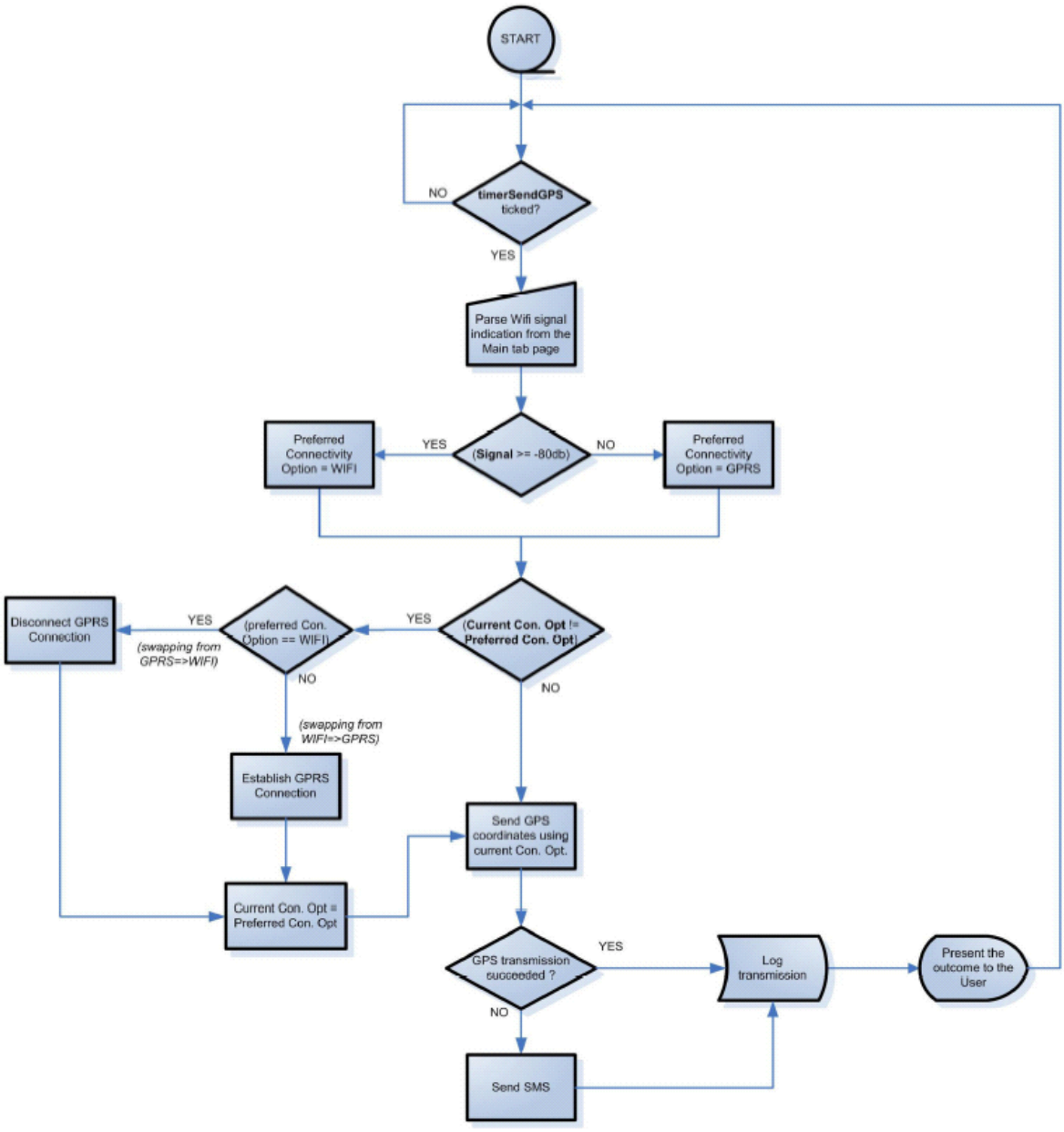

Figure 11 Flow Chart for the Transmission of GPS Coordinates

The second timer, `timerSendGPS`, is responsible for sending the GPS Coordinates. It is initialised when the user starts the communication, it ticks every 15 seconds and sends the current GPS coordinates to the server. Its function, described in the flow chart of Figure 11, is mainly achieved with the following actions in order:

1. Parses the current WiFi signal strength from the field of the Main tab page that has been updated from the timer `WifiSignalStrength`. If this value is higher than -80db then it means that the WiFi signal strength is considered at least good enough to use and the preferred connectivity option is set to WiFi. If the WiFi signal value is lower than -80db or has not been set, then the WiFi coverage is either poor or non-existent and the preferred connectivity option is set to GPRS.
2. If the preferred connectivity option is different to the one currently being used, the application swaps to the other IP-based connectivity option. If the application has to swap from WiFi to GPRS then it first tries to establish a GPRS connection based on the reference that it has for that connection and then swaps to it. If the application has to swap from GPRS to WiFi then the GPRS connection is disconnected[1] and the application swaps to WiFi.
3. The application tries to send GPS coordinates with the use of the `IPConnector` object over the preferred connectivity option. If this transmission fails, then the application swaps to the backup alternative option, which is SMS over GSM, and therefore tries to send the coordinates via SMS.

The aforementioned high-level procedure is executed every 15 seconds, which is the default transmission interval for the application to send GPS coordinates. Due to hardware and operating system restrictions, this interval might not be followed precisely, mainly in the cases of swapping from one connectivity option to another. For example, when the application swaps from WiFi to GPRS it has to wait for the GPRS connection to be fully established before it tries to send GPS coordinates.

4.2. GPSConnector

The `GPSConnector` class is used to represent the connection with the GPS module (either internal or external). An object of the `GPSConnector` class is created on the `GUI` class so that the GUI class will have a reference to that GPS connection. The `GPSConnector` object is mainly responsible for continuously obtaining data from the GPS module and passing it as NMEA phrases to the object of the `NMEAInterpreter` class.

Consequently, the `NMEAInterpreter` object parses the received data and triggers the appropriate events upon the validity of the data. These events are handled by EventHandlers of the GUI, which are responsible for updating the GUI Form that is presented to the rescue worker and some global variables that are needed for the main functionality of the `GUI` class. The high-precision `NMEAInterpreter` implementation used is written by Jon Person (author of the "GPS.NET") albeit slightly modified to make it more stable in our context.

In more detail, the `GPSConnector` object retrieves as an argument from the `GUI` object the GPS module that will be used, internal or external, and tries to initiate a connection with it. If the internal device is used, then by trying to poll the COM7 serial port of the device at 56kbps baud rate it will initiate the GPS integrated module of the device. If the external device is used, a polling procedure on the COM6 serial port will pop-up the Bluetooth console to initiate the connection with the external device. When the serial communication is successfully established, an `NMEAInterpreter` object is created and all the

[1] Based on Operating System restrictions that the iPAQ device has, the GPRS connection cannot remain active in the background while a WiFi connection is active. Instead, it has to be disconnected so that the device is able to use WiFi connectivity.

appropriate `EventListeners` are assigned to events that could be fired from the `NMEAInterpreter` object. The serial connection is passed as an argument to a new thread that will continuously retrieve data from that connection. When all the sanity checks for the retrieved data are successful then these are passed to the `NMEAInterpreter` object for parsing.

Note that the NMEA format is the usual format used from GPS devices to report the data that they have received from the satellites. According to the NMEA protocol, there are 62 defined code phrases with an average of 15 arguments each. These include data from satellites varying from GPS almanac data, to wind direction, water speed and satellites' time [12]. Most of the GPS devices that are relatively cheap to purchase could interpret the 19 most useful and frequently used NMEA phrases that are presented in Table 3. Table 3 also includes in bold the four most significant NMEA phrases that the `NMEAInterpreter` object parses.

Table 3 Most frequent NMEA phrases used (adapted from [13])

NMEA Code	Description
$GPBOD	Bearing, origin to destination
$GPBWC	Bearing and distance to waypoint, great circle
$GPGGA	**Global Positioning System Fix Data**
$GPGLL	Geographic position, latitude/longitude
$GPGSA	**GPS DOP and active satellites**
$GPGSV	**GPS Satellites in view**
$GPHDT	Heading, True
$GPR00	List of waypoints in currently active route
$GPRMA	Recommended minimum specific Loran
$GPRMB	Recommended minimum navigation info
$GPRMC	**Recommended minimum specific GPS/Transit data**
$GPRTE	Routes
$GPRTRF	Transit Fix Data
$GPSTN	Multiple Data ID
$GPVBW	Dual Ground / Water Speed
$GPVTG	Track made good and ground speed
$GPXTE	Cross
$GPZDA	Date and Time

When the `NMEAInterpreter` object parses a known phrase, it triggers the suitable events, which execute the appropriate EventHandlers. These EventHandlers are responsible for updating the `GUI` main class and global variables that are stored for the main functionality of the application.

4.3. IPConnector

The `IPConnector` object that the `GUI` class holds represents the IP based connectivity option that is used at a specific time. The server application listens for IP connections from the client using either WiFi or GPRS. The `IPConnector` class can facilitate both IPv6 and IPv4 connections.

Upon the creation of the object of the `IPConnector` class, a socket is opened with the server based on the IP address that the `GUI` class passes as argument to the constructor of the `IPConnector` class. Consequently, this object is used to send GPS coordinates to the server over the instantiated socket and is responsible for informing the `GUI` class if the coordinates have been sent successfully. In addition, if there has been a failure in the transmission of the GPS coordinates, meaning that the socket was broken, then this class uses a background thread to force reconnection with the server and the creation of a new socket for future use under a specific connectivity option. Finally, this class also informs the Main tab of the GUI Form for network related issues, such as the IP address of the client, failures in obtaining a socket or failures in the reconnection procedure.

4.4. Map

Although presenting a map to a rescue worker is not a critical requirement, it was decided that it would be a useful feature, especially for demonstration purposes. Thus, the Map tab shows the rescue worker his/her current location as determined by the client application.

The Map tab was found to be valuable during the testing of the software, because it enabled us to correlate the GPS coordinates being sent to the server without having to be present at the server location.

The Map functionality is implemented using the Google Maps API v2.0 [14]. A html page containing javascript is created and is placed on a specific webspace. When the rescue worker opens the Map tab and taps the appropriate button (Figure 10), the client retrieves the current latitude and longitude and converts them into their decimal equivalent. This procedure is mandatory because the Google Maps API cannot parse latitude and longitude in the NMEA format.

Consequently, a request is initiated for the described html file, passing as arguments the latitude and longitude in decimal format along with the node ID of the client that makes the request. This request is performed over the current IP-based connectivity option. The online file parses the arguments of the request and embeds a Map with the region that corresponds to the coordinates. A “balloon” marker is placed on the exact GPS coordinates that the client sent in the request. If not already active, a browser is loaded on the client side presenting the map with the marker plotting the location of the rescue worker (Figure 12).

Figure 12 Loading the Minimo web browser (left) and the resulting map (right)

Although the Map tab is used from the client application, it is not truly integrated since it launches an external web browser that makes the request for the map. Unfortunately, the `webBrowser` container that the .NET Compact Framework provides uses the default web browser of the Operating System, namely the Pocket Internet Explorer 3.0. This version of Pocket Internet Explorer supports a very limited set of javascript that cannot provide the functionality demanded by the Google Maps API. Thus, the Minimo web browser, developed by the Mozilla Corporation, is used. It should be mentioned that Minimo, which is currently the only free web browser for mobile devices that fully supports javascript, uses a significant amount of the resources on the PDA. Therefore, mapping the location of the rescue worker is only activated on demand rather than by default.

4.5. Logging

The client includes a powerful logging capability that appends to a log file a considerable information regarding user input, data retrieved from the satellites and network status along with a timestamp. This log file is updated with information regarding the current connectivity option used, the payload of the packets sent, the IP address of the client, the WiFi signal strength at specified intervals and many more useful pieces of information.

The logging capability has been implemented in such a way so that data can be examined offline. This forms the basis that allows missions to be 'replayed' after they occur, allowing the mountain rescue team to analyse the performance of the team during that mission.

Since the log feature of the client records the available connectivity options and network status that were observed at specific coordinates, this information can be used to aid in the preparation of future missions. For example, over time persistent communication black spots (for any connectivity option) can be identified and anticipated in advance. WiFi antennae located with rescue vehicles could be repositioned or temporary, lightweight relays, which can be carried in backpacks, could be deployed to overcome the black spot.

4.6. Hardware

The choice of hardware for the LAS client device was based on many factors, including the requirements discussed in chapter 2, in addition to the needs of the LAS system architecture itself. However, the LAS client device can be summarised as needing to posses the follow attributes:

- Small
- Lightweight
- Waterproof
- Durable, shock resistant
- Battery runtime of 4 hours
- 802.11 (b and g)
- GSM/GPRS/EDGE
- Internal GPS
- Bluetooth

Considering that the LAS client device would be carried by rescue workers, ideally the device would also be wearable, have an adequately sized screen and have large buttons that can be operated by a user wearing gloves. However, if the client application can be left to operate without intervention, the need for a large screen and large buttons is somewhat lessened. Where possible the client device should have a battery runtime of 4 hours or more, with the LAS client running, as this should suffice for the majority of search and rescue operations. Battery performance should not be drastically affected by cold weather although some drop in efficiency is to be expected. By evaluating and prioritizing the above requirements it was decided that a PDA-like device was needed. A notebook solution was considered too cumbersome for a rescue worker to carry and a no suitable smartphone solution was found. Smartphone solutions that were considered were generally found wanting in connectivity options, GPS or a suitable development platform. The client device must be able to utilize the defined connectivity components (as discussed in section 3.2) and thus it must have WiFi, GPRS and GSM capabilities. Ideally, the device should have an internal GPS receiver so that less equipment needs to be carried. However, connecting to a small form-factor external GPS device via Bluetooth was deemed acceptable, albeit not ideal.

The Network Mobility group at Lancaster University already had at its disposal a HP iPAQ 6315 PDA and a Pretec GPS device that, at first glance, could facilitate the needs of the LAS client. The HP iPAQ 6315 PDA device has a 3.5inch screen, WiFi, GPRS and GSM capabilities and could establish a Bluetooth connection with the Pretec GPS device in order to receive GPS coordinates over a serial communication. The Pretec GPS device is small, lightweight, can concurrently monitor 12 satellites and Bluetooth capable.

However, the outcome of this period defined that although the iPAQ 6315 PDA suited the needs of the LAS client in terms of its hardware capabilities, its Operating System and available programming framework were found to be inadequate. In particular, the iPAQ 6315 runs Windows Mobile 2003 and the available SDK for developing an application such as the LAS client was limited (this is analysed more thoroughly in the following section) and thus new PDA devices were purchased.

Figure 13 IPAQ 6915

The purchased devices were the HP iPAQ 6915 (Figure 13) offering new features and providing greater flexibility for developing an application to facilitate the transmission of the GPS coordinates. The iPAQ 6915 runs Windows Mobile 2005 and offers an API with greater potential than the previous iPAQ. Moreover, the iPAQ 6915, has an internal integrated GPS receiver that canobtain GPS coordinates, a feature that eliminates the need for the external GPS receiver in a mountain rescue mission.

Table 4 Features of the IPAQ 6915

Feature	Detail
Operating System	Microsoft® Windows Mobile™ 5.0, Phone Edition with Messaging and Security Feature Pack 5.0
Processor	Marvell PXA270 CPU 312 MHz
Memory	64-MB SDRAM (45 MB user accessible)
Display	3.0 in (75 mm) diagonal, transflective TFT color, 240 x 240 pixels, 0.24mm dot pitch, 64K-color support, touch screen
Connectivity	Integrated Quad band GSM/GPRS/EDGE wireless radio with automatic band transition; Integrated Wi-Fi (802.11b); Integrated Bluetooth 1.2 wireless technology; Integrated IrDA SIR
Dimensions (w x d x h)	2.8 x 0.71 x 4.65 in (71 x 18 x 118 mm)
Weight	179.45 g
Battery	Removable/rechargeable 1200 mAh Lithium-ion
Extras	Integrated GPS, Built-in VGA Camera

Table 4 provides a summary of the features and capabilities of the HP iPAQ 6915.

4.7. Software

Developing an application such as the LAS client for a PDA does not inherit a wide range of options regarding the available programming languages and SDKs, especially when there are requirements to use numerous network connectivity options and technologies. The requirements of the LAS client demand full access to the WiFi and GSM/GPRS network adapters, to the Bluetooth stack, as well as being able to create and manage sockets for both IPv4 and IPv6.

Moreover, since PDAs have limited CPU, memory, storage and display capabilities, the chosen programming language and development platform must be able to comply with their needs. Finally, although presenting a GUI to the rescue worker is not a critical requirement, an API with display capabilities in such devices is desirable for training rescue workers, debugging problems and for demonstration purposes. Research based on these requirements defined that the two most suitable and programming languages and environments for developing the LAS client for a PDA are Java and Visual C#.

Table 5 Various Java solutions for Pocket PC (modified from [20])

	Sun Personal Java	Esmertec Jeode	IBM Websphere Studio	Blackdown J2RE (ARM Port)
Supported OS	WinCE 2.11	WinCE 2.11, Pocket PC, Linux	WinCE 2.11, Pocket PC	Linux
JVM Compatibilty	Personal Java compliant	Personal Java compliant	J2ME MIDP 1.0	Java 2
Speed	Fast	Fast	Fast (ahead of compile time)	Slow start, reasonable execution
Supported Hardware	MIPS, SH3	Dell Axim X5	PocketPC, WinCE, PalmOS, Windows	iPAQ H36xx (or above)
Cost	Free	$49.95	Depends on the micro environment used.	Free

Starting with Java, Sun identified that there was a need for providing a subset of J2SE (Java 2 standard edition) for mobile devices and initially published Sun Personal Java, which was later renamed to J2ME and then to JavaME (Java Micro Edition).

JavaME successfully dealt with the constraints the small devices present and tried to support Sun's motto of *"write once, run everywhere"*. However, quoting from Sun's web portal, *"Java ME platform is a collection of technologies and specifications that can be combined to construct a complete Java runtime environment to fit the requirements of a particular device or market"* [19]. Therefore, this runtime environment is completely device-dependant and requires specific configurations and profiles not only for each device but also for each implementation of every feature on a device. This is a possible reason why Sun does not provide any official JVM (Java Virtual Machine) support for mobile phones, Pocket PCs and PDAs. Consequently, the evolution of JavaME is based on various virtual machines, configurations

and profiles that many third party companies have developed to either support their mobile devices (e.g. Nokia phones [21]), or specific implementations of a feature such as the Bluetooth stack v2.0 or the Wireless Industry specification (JTWI) [22]. Table 5 presents only four significant Java based solutions that could be used for developing an application for a mobile device, which illustrates the variety of properties that must be considered[1].

The major alternative programming language that could be used to develop the LAS client application for the PDA is Visual C#. Microsoft Visual .NET framework was created to provide programmers with a variety of libraries that could be used to easily create new applications, especially when combined with Visual C# [24]. Emulating Sun, Microsoft has also defined a subset of the .NET programming framework for mobile devices called the .NET Compact Framework (.NET CF) which includes libraries that are specially tweaked for use in devices with limited resources [25]. Although .NET CF can be used in devices that run only Microsoft's Operating Systems, it provides a significant advantage in that unmanaged code can be called from managed code in such devices. In other words, unmanaged code such as C or C++ can be used within the .NET CF environment to have access to any native DLL files that the operating system uses. This allows managed code such as C# to potentially have access to all the hardware and software features that a mobile device has.

Comparing the different options for the programming language and SDK that could be used for developing the LAS client application, it is clear that the combination of Visual C# and the .NET Compact Framework holds significant advantages over Java based solutions. However, even with this combination there are limitations. The .NET CF has seriously restricted capabilities compared to the .NET framework used on PCs and lacks essential functionality that one might expect to see in a development framework. On the other hand, the benefit of having access to native DLL files and embedding C/C++ code to gain access to system resources presented an advantage that could not be disregarded. Thus, it was decided to use Visual C# and .NET CF to develop the LAS client application.

4.8. Main Difficulties and Restrictions

This section presents the main restrictions that we observed with our chosen client device. We have partitioned them into software and hardware restrictions that significantly affect the client LAS implementation.

4.8.1. Software Restrictions

The initial design of the system included two different classes, namely `WIFIConnector` and `GPRSConnector`, to provide connectivity for the WiFi and the GPRS connectivity options, respectively. The underlying idea was that each of them would communicate with the server via their own IP socket. This way, when there was an indication of poor WiFi signal strength, the `GPRSConnector` would establish a GPRS connection, acquire a socket with the server and be on stand-by when the WiFi signal would be lost completely. Likewise, if the GPRS connectivity option was active when sufficient WiFi signal was detected, the `WIFIConnector` would initially try to acquire the socket with the server and then, if successful, instruct the `GUI` class to swap transmissions of the GPS coordinates to the WiFi option.

Unfortunately, from the application's point of view, Windows Mobile 5.0 (and indeed, all of Microsoft's mobile Operating Systems that are based on the CE platform), uses two different network components that are included with dynamic library of the Operating System to expose access to network resources. The first component is the Connection Manager that, according to the Windows CE Networking Team

[1] Two extensive comparisons of the many Java based solutions for PDAs can be found in [20] and [23].

Weblog [15], "manages all network resources" and claims to "provide access to different destination networks". The second component is the Connection Planer that is "responsible for choosing one or a set of connections that will satisfy a connection request" [16]. To put it simply, in theory, if an application demands a connection to the Internet, the Connection Planer will evaluate all the predefined connection settings that the device has and instruct the Connection Manager to initiate a certain connection to facilitate the application's request.

Unfortunately, in practice there are times when the Connection Manager will not follow the Connection Planner's "proposal" based on some specific criteria. Again, quoting again from Microsoft Windows CE networking team's blog, Connection Manager might "restrict network resources to different connection requests from applications based on their priority and resources' security properties". For example, if both a WiFi and a GPRS connection are up and the WiFi connection is in use by an application, the Connection Manager will not swap to the GPRS connection even if the Connection Planner might instruct it to do so. One reason for this is that a WiFi connection generally provides higher data rates than a GPRS connection and, using this logic, the Connection Manager will not swap from WiFi to GPRS when the Connection Planner instructs it to do so if there is sufficient WiFi signal. While this may be a prudent course of action in some cases, it is naïve to apply this logic for all situations. The WiFi signal seen by the PDA may be sufficient, but other factors can cause problems with Internet connectivity. For example, the WiFi network may be severely congested or there may be routing problems at the egress of the WiFi access network.

In addition to this, there are occasions when the Connection Manager will not immediately swap from a GPRS connection to a WiFi connection. This is based on the fact that a GPRS connection takes a relatively long time to establish and therefore the Connection Manager waits for several seconds to swap from GPRS to WiFi in order to avoid connection 'flapping', which could leave the device without any useful connectivity. Furthermore, the Connection Manager cannot support packets being sent and received from more than one active IP-based connection simultaneously. Even if two or more IP-based connections are active and available to use, packets can only be transmitted over one of them.

Therefore, due to the restrictions of the Connection Manager, our initial design for having two different classes (`WIFIConnector` and `GPRSConnector`), one active and one on stand by, was replaced by the `IPConnector` class described above.

For our scenario in the Mountain Rescue Domain, and in order to be able to swap from WiFi to GPRS a hybrid implementation was used to access functions in the coredll.dll and cellcore.dll libraries. The implementation developed for the client side overcomes the deficiencies of the Connection Planner and directly instructs the Connection Manager which specific network it should use[1]. Unfortunately, the Connection Manager still exhibits problematic behaviour.

There are two apparent swapping procedures, namely swapping from WiFi to GPRS and swapping from GPRS to WiFi. In the first case, although the application is able to identify poor WiFi signal strength, establish a GPRS connection (handled as a dial up connection with the telecommunication provider) and instruct the Connection Manager to swap to it, the Connection Manager does not follow this instruction until it loses the WiFi signal completely. This functionality increases the possibility of losing packets transmitted over a WiFi socket when in fact the WiFi signal is too low. In the second case, swapping from GPRS to WiFi, although a WiFi network might be available, the Connection Manager does not immediately follow the instruction to swap to the WiFi network if the GPRS connection is active. To confront this inefficiency, the following workaround was implemented.

[1] This was developed with the use of a library from OPENNETCF Corporation and by invoking functions to native DLL files.

When the application needs to swap from GPRS to WiFi, the GPRS connection is forcibly shut down, with the aid of the coredll.dll library, so that the Connection Manager has no options but to swap to the WiFi network.

4.8.1.1 IPv6

In order to utilise IPv6 in the purchased iPAQ 6915, both the Operating System of the device and the programming framework should be able to support IPv6. As such, both Windows Mobile 5.0 and the .NET Compact Framework, which was used for the development of the client's application, can theoretically support IPv6.

Unfortunately, Windows Mobile 5.0 do not provide neither a GUI form for configuring IPv6 nor the usual ipv6.exe tool to manually configure IPv6 on the PDA.

Moreover, the IPv6 address of the iPAQ's WiFi adapter could not be found either programmatically or within the registry. After many experiments trying to configure the IPv6 feature on the device a workaround[1] was used to identify the link-local address that the device obtains during the autoconfiguration and the neighbour discovery phases. By using the link-local IPv6 address of the PDA we were able to establish a communication with the server by using IPv6 TCP sockets over the WiFi network. Unfortunately, using a link-local (rather than a global) address, means that the server needs to be on the same link as the client. Clearly, this is an unacceptable situation for a real deployment.

The first testbed we used during the development of the applications for the LAS included an ad-hoc wireless network between a laptop configured with IPv6 and running the server application and the iPAQ 6915 running the client application. Consequently, the transmission of IPv6 packets over a WiFi network was successfully implemented and tested on this testbed in conjunction with the SMS over GSM connectivity option.

For GPRS connections, there is no IPv6 capability provided by any of the major UK telecommunications providers. Thus, all IP connections over GPRS are IPv4 and, in many cases, also use NAT so that the client device (mobile phone or PDA) only sees a private address in the 10.0.0.0/8, 172.16.0.0/12, or 192.168.0.0/16 ranges. For a client application to use IPv6 with GPRS, an IPv4/IPv6 transitioning tool that can work across IPv4 NATs is necessary. One such tool is Teredo [17], although we have not yet tested the LAS using Teredo.

In the process of developing the functionality to utilize the GPRS connectivity option it has was decided that the server machine should be 'dual-stack' i.e. have a global IPv4 address so that it can be addressed from clients over the GPRS network and an IPv6 address so that it could be addressed from clients via WiFi.

4.8.2. Hardware Restrictions

The iPAQ 6915 is equipped with a GPRS class B module, which has some restrictions in terms of connectivity [18]. Class B GPRS modules can be connected to both GPRS and GSM services but are only able to use only one service at any given time. When a request is made for one service, the other service is temporarily suspended until the requested service is fully completed. For the LAS client, this limitation presented significant restrictions in the two following cases.

1. When the application was able to identify that the WiFi signal strength was poor or zero, it would initiate a GPRS connection. However, the GPRS connection needs at least 10 seconds to be established and in addition, the application should first acquire a TCP socket and then swap to

[1] The only way of identifying the IPv6 address of the device was by capturing WIFI packets when the wireless network adapter of the PDA was turned on and thus notice the IPv6 address that the device was publishing as part of the network discovery phase.

GPRS. Many tests during the development phase proved that if the client was sending SMS messages during the procedure of establishing a GPRS connection, the GPRS connection could not be set up at all because the GPRS module was suspended during the transmission of SMS messages. The workaround that was found in order to initiate the GPRS connection was to wait until the connection is established before sending SMS messages. The impact of this workaround was that the application was not able to send GPS coordinates in fixed intervals but was, at least, making the GPRS connection available for future use when needed.

2. When there is no WiFi connectivity, the client will send SMS messages until the TCP socket over GPRS has been established. However, the GPRS module is suspended upon every SMS transmission; this adversely affects the time taken to establish the TCP socket and in some cases the socket creation will even time out.

5. LAS Server Implementation

The server component of Figure 3 includes the server application running on a server machine located at the mountain rescue team's HQ. The main aim of the server application is to listen for incoming messages that are sent from the LAS clients regardless of the connectivity option used for their transmission. The server application processes the payload of each arrived message and plots the received GPS coordinates of the rescue members onto a map to assist the mission coordinator in tracking the progress of a mission and take informed decisions.

The server application is mainly implemented in Visual C# using the .NET framework v2.0 and runs on the mountain rescue team's server located at the HQ in Cockermouth (Cumbria, UK). It is a multi-threaded application listening for TCP connections from many clients coming in over its connection to the CLEO network[1]. In addition, the server application has a GSM-enabled modem, which receives all the SMS messages that are sent from the clients and forwards them to the server application for processing.

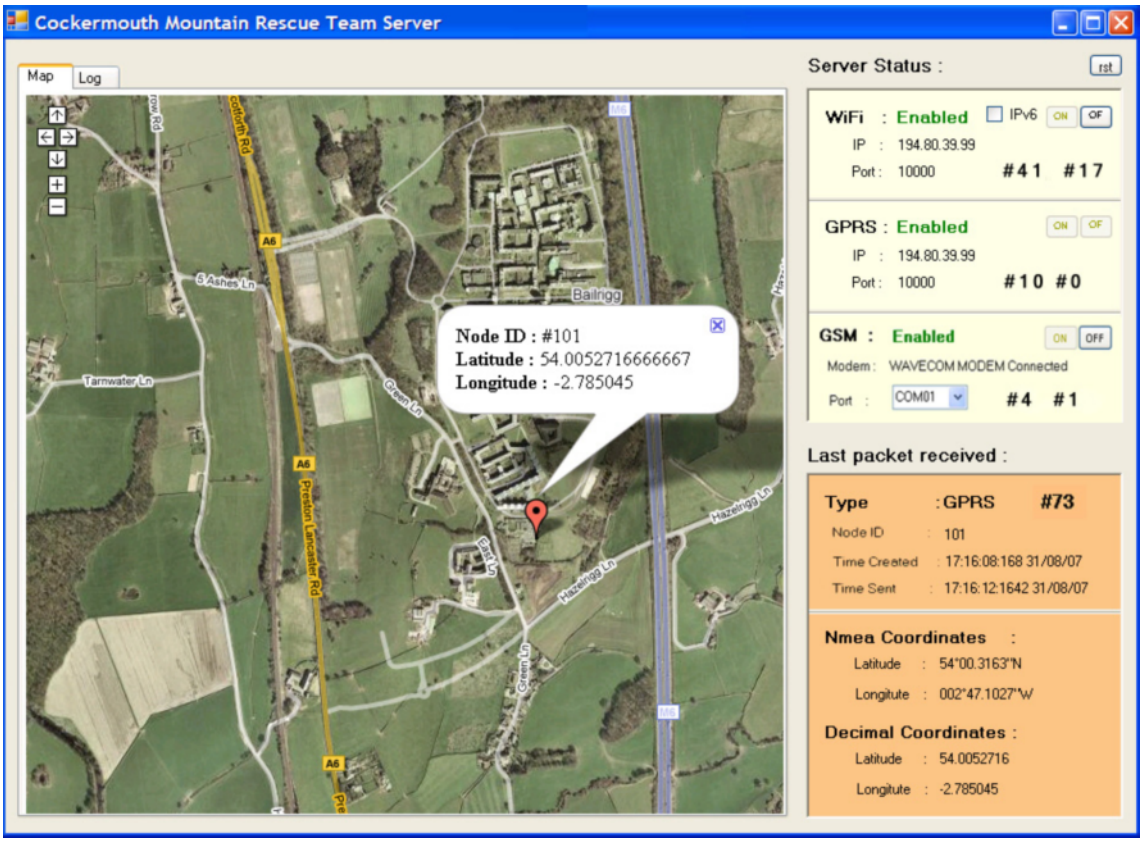

Figure 14 The LAS Server Application

[1] CLEO stands for Cumbria and Lancashire Education Online and is a MAN managed by Lancaster University. The CLEO network connects all the schools, colleges and universities in the counties of Cumbria and Lancashire to JANET. For information regarding network connectivity between the mountains and the team's HQ, please refer to [4] and [5].

An important part of the server application is to present the location of the rescue workers in a clear and coherent way on suitable maps. Figure 14 shows a screenshot of the LAS server application, which is divided into three distinctive parts; the Map, the Server Status and information for the last message received. The map functionality is currently implemented in two ways. The first uses a javascript file that overlays useful information from the received messages onto a map using the Google Maps API. Alternatively, the same positioning information can be overlaid onto more detailed Ordnance Survey maps. Either way, the mission coordinator has the ability to see more information about the last packet/message received either by clicking on the point on the map or by examining the right bottom corner of the server application. Information about the received coordinates in NMEA and decimal format, as well as the connectivity option used for the transmission of a message and timestamps for its creation and transmission are also available. Information about the current status of the server, the IP address/port that the server listens to, the overall status of the listening functionalities of the server and counters for the messages received are also displayed at the top right corner of the application.

The map functionality can be present as a tab within the LAS server and/or embedded with the general Mountain Rescue software that incorporates knowledge databases and search theory algorithms[1]. An example of the map functionality used with the Mountain Rescue software and using both Ordnance Survey maps and Google Map API is shown in Figure 15.

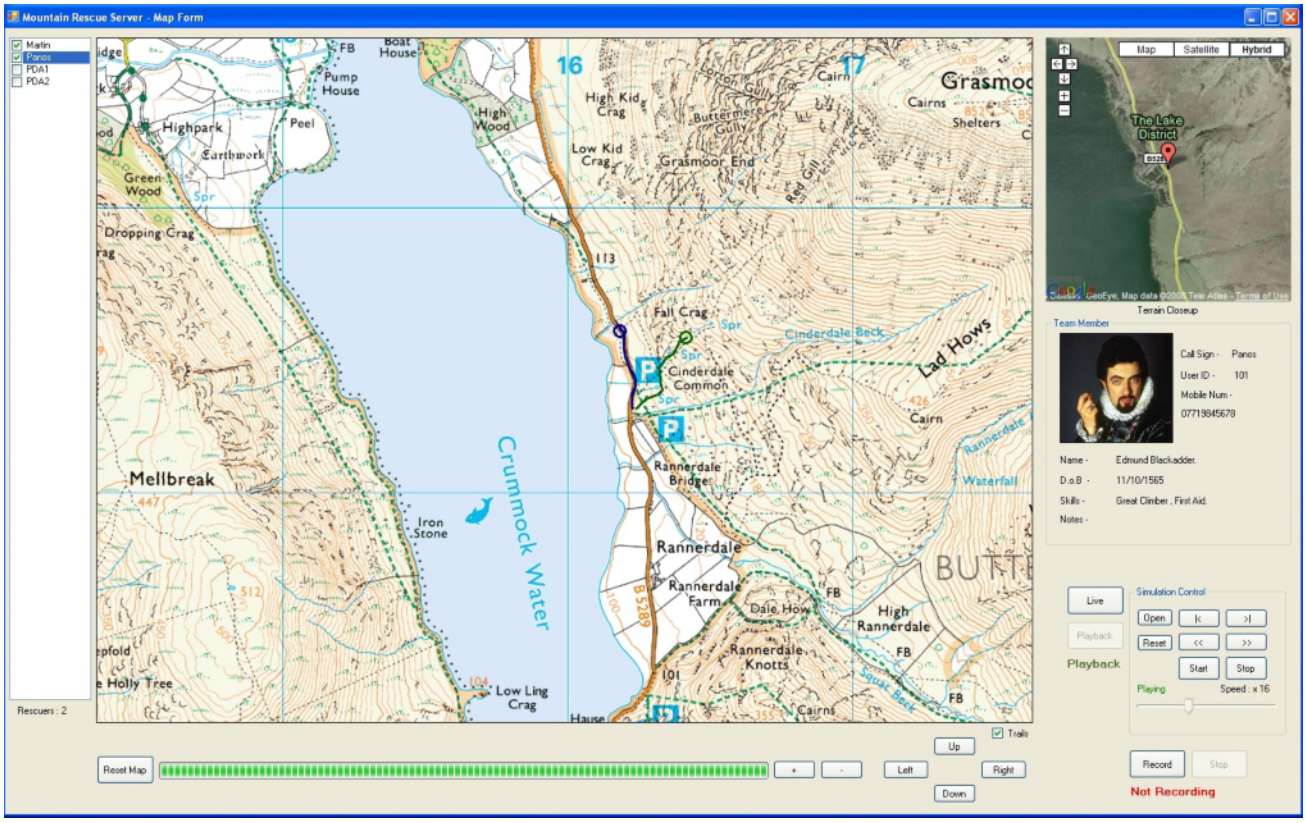

Figure 15 The LAS Server used with the Mountain Rescue Software

[1] For more information on this software please consult [4]

The server application imposes important procedures on the incoming messages. As every message follows the defined unified message format (UMF) of the communication framework, the server application can identify the person whose device has sent a message from the node ID field and also authenticate it using the node ID field and the Security Code. Complementing TCP's ability to rearrange out-of-sequence IP packets at lower levels of the networking stack, the server application also has the ability to rearrange out-of-sequence messages at the application layer using the node ID and the Sequence Number fields. Such a rearrangement can be valuable when for example a client has sent an SMS message and then regained WiFi connectivity and sends another GPS update. It is possible that the WiFi message will arrive earlier than the SMS message and thus the server has to filter the information that it displays to the mission coordinator. Filtering also applies when the server application receives offline messages that are depicted with a different colour since these messages contain location points that rescue members have been in the past. Identifying, filtering and storing information from such messages is valuable for off-line studies of the movement of the team and for enabling us to improve and apply search theory techniques.

The remainder of this chapter will describe the GUI of the server and its main functionality in section 5.1, the GSM module in 5.2, the Map functionality in section 5.3, its detailed log file in section 5.4, with sections 5.5 and 5.6 describing the hardware and software respectively. Finally, section 5.7 describes the scenario for integrating the LAS with an alerting solution.

5.1. Main Form Class

The `MainForm` class of the server is composed of two distinct logical and practical parts, similar to those that client has, developed with the partial class implementation feature of Visual C#. The first part is the `MainForm.Designer.cs` partial class, which holds all the GUI elements of the form that the rescue coordinator notices on the server application and handles all the interaction with the user. The second part is the partial `MainForm` class that includes the main functionality of the server. The two aforementioned parts are going to be described in the following sections.

5.1.1. GUI

Figure 14 presents the GUI of the server application. The `MainForm.Designer` partial class includes a main central area presenting two tab pages, the Map and the Log, and also two panels on the right side, presenting the server status and information for the last message received.

Starting from the tab pages of the main central area, the Map tab page presents a map and a marker on the exact location of the client that sent the last message. If the marker is clicked then the exact longitude, latitude and the node ID of the client that sent the message is presented on the screen (this is illustrated in Figure 14 and the same functionality shown to the right of the map in Figure 15). The Log tab page (Figure 16) presents an extensive Log containing information regarding all the functionalities of the server and considerable information for all the received messages.

The rescue mission coordinator monitors the Map tab along with the two side panels that present all the information needed for a mission. The panel that is located on the upper right side of the server application presents the status of the server regarding the three different connectivity options that are supported. Therefore, the mission coordinator can see which of them are enabled and the server currently monitors for messages. Appropriate buttons are also presented on the panel so that the rescue coordinator can turn the listening functions ON and OFF accordingly. In addition, the server status panel presents a counter for each connectivity option so that the coordinator can be aware of the number of messages received from each connectivity option. This panel also presents information such as the IP address and

port number that the server is using for listening for IP packets and the serial port that the server uses to communicate with the GSM module.

The panel that is located in the lower right corner of the GUI presents information for the last message received. In detail, it displays the connectivity option that was used to send the last received message, the node ID of the client that sent it, the time that the message was sent by the client and the time it was received by the server. Furthermore, the coordinates of the message that has just been received are presented in both the NMEA and decimal format. Finally, a counter on the upper right corner of this panel presents the total number of messages received.

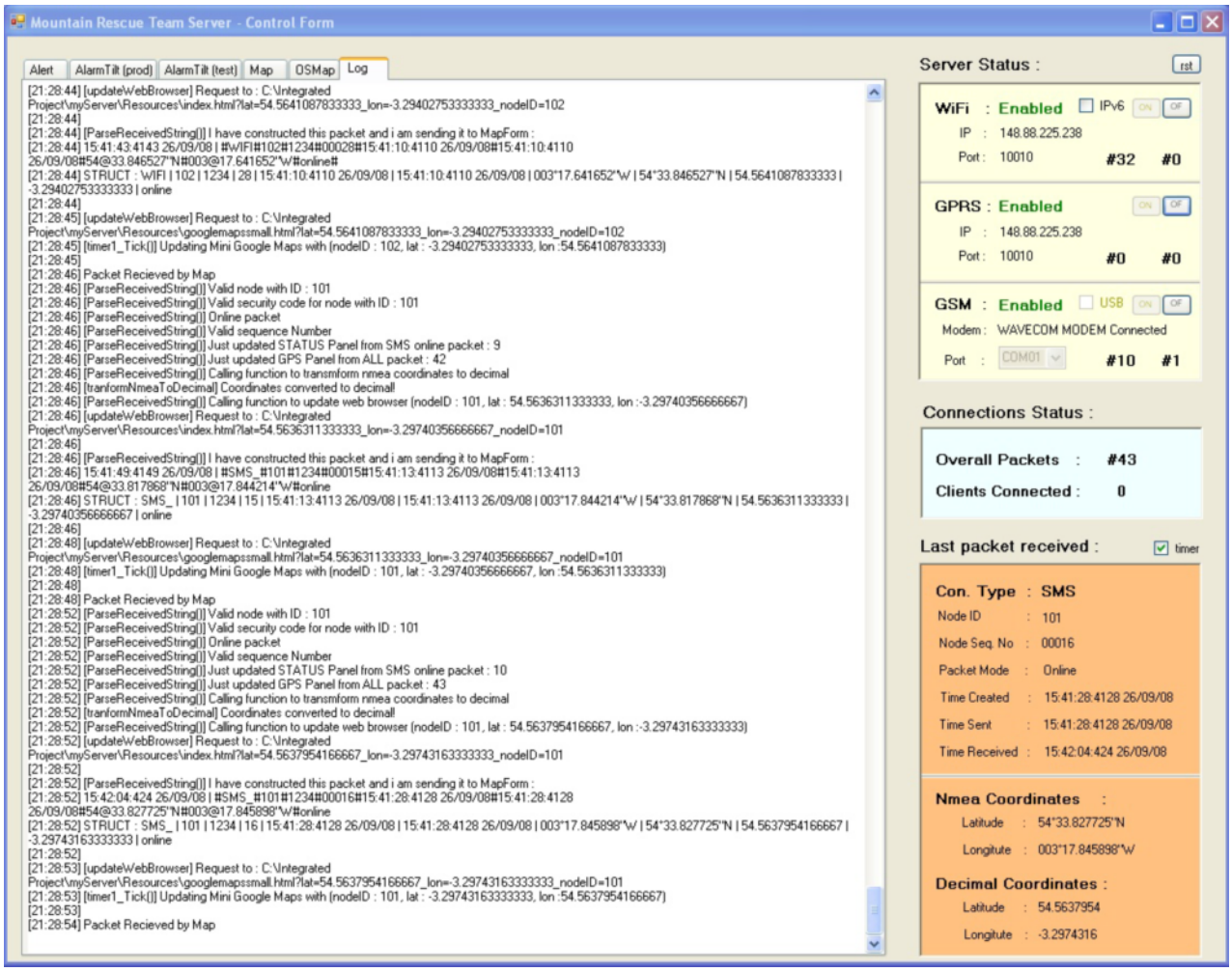

Figure 16 The server application logging tab

5.1.2. Main Functionality

The server application starts listening for messages sent from clients when the rescue coordinator enables the appropriate listening functions. Therefore, when a user clicks the "ON" button of the WiFi connectivity option (Figure 16) the server creates a socket bound to its global IPv4/IPv6 address and listens for packets on the configured port number. This user's action also enables the GPRS module as the server listens only to one IP address for packets sent using either IP based connectivity option. When the user enables the GSM module then a serial communication is initiated to the installed GSM modem

Regarding the implementation for listening to IP packets, the server uses a separate thread to listen for clients that try to acquire a TCP socket and initiate a connection. When such a request is identified, it is passed over a different thread to facilitate the connection with that client and receive data over the instantiated socket.

Therefore, the initial thread is able to listen for more clients' requests. The thread that holds the socket with a client continuously retrieves packets from it and passes them to a specific method, entitled as `ParseReceivedString(…)`, which parses the payload of the packets retrieved. This method is used for every received message (even for SMS messages) as every message has a defined uniform format.

Initially, the `ParseReceivedString(…)` method splits the received payload into "words", based on the delimiter character "#", and performs numerous checks to verify the validity of the retrieved data. In brief the following actions are performed in order :

1. Check if every word has a valid format based on the defined length and on the fact that some words should include some special characters. For example the longitude and latitude values should include the character "@".
2. Check if the node ID and the security code of the client that sent the message are valid.
3. Check if the sequence number of the message received is valid.
4. Identify the connectivity option that was used and update the suitable counter.
5. Update the panel that holds the information for the last retrieved data (node ID, time sent, time received, GPS coordinates in NMEA format)
6. Call a function to transform the GPS coordinates from NMEA to decimal format.
7. Check that the converted decimal coordinates correspond roughly to a position in the UK.
8. If the coordinates are not the same with the ones previously retrieved then a request is made to update the Map on the suitable tab page.

The above procedure is done for every packet that is retrieved from a socket until the client gets disconnected. If an SMS message is retrieved (discussed in the following section) the `ParseReceivedString(…)` method is also called with the payload of the SMS as an argument. Generally, the server application continuously listens for incoming packets until its functions are disabled by clicking the OFF buttons for the appropriate functionality.

5.2. GSM Module

The LAS server is able to listen to SMS messages by monitoring the SMS messages retrieved via a GSM modem, either connected to a serial port or using a Bluetooth connection. This functionality is written with the aid of the GSMComm library written by Stefan Mayr [26].

Before the server can initiate the GSM functionality, both the GSM modem and the server should have a Bluetooth association or, alternatively be connected to via the server's serial port.. When the user enables the GSM functionality on the server's application, a serial connection is created (over Bluetooth also) between the GSM modem and the server. From this point on, the server application is able to query the GSM modem with an AT command set that is exposed over the serial connection.

When the serial connection is initiated the server application assigns a few `EventListeners` to listen for events that are triggered based on the status of the GSM modem. Moreover, the server application enables a functionality that forwards every arrived SMS message from the GSM mode to the server and deletes the message from GSM modem. The most significant `EventListener` that is assigned for the GSM functionality is the `comm_MessageReceived(…)` that is triggered when a new SMS message is

forwarded to the server application. This method decodes the received SMS to an SMS PDU and strips the actual data that the SMS carries (the text message) from all the other information. Consequently, all the information regarding the sender of the SMS and the time sent are presented on the Log tab page, and the actual string is passed as an argument to the `ParseReceivedString(…)` method that is responsible for analysing content of the message as was previously described.

5.3. Map

The Map tab page, illustrated in Figure 14, is responsible for presenting a Map to the rescue mission coordinator and is automatically updated when new GPS coordinates arrive, regardless the connectivity option that was used for their transmission. However, there are three distinct occasions when the server application does not automatically update the map, namely:

1. When the GPS coordinates are composed of zeros.
2. When the GPS coordinates do not correspond to a sensible location point, indicating an error with the client GPS receiver. How we define a 'sensible' location point is rather subjective but the rationale is that the location must be in the general search area and must not be an iompossible distance from the previous location (e.g. it is impossible for a rescue worker to walk several Km in only 30 seconds).
3. When they have not changed from the previously received coordinates sent from the same client.

The Map tab page includes a `webBrowser` container that acts much like the container described for the LAS client application, with the only difference being that the functionality for the server side is completely integrated into the LAS server application. Furthermore, the entire functionality of the Map tab can be embedded into the more general monitoring and management software used for the Mountain Rescue team.

5.4. Log

The server application includes two different types of logging functionality, namely the Log tab page and a log file. Both logging features are almost identical with some exceptions when the server application starts and closes because at that time the log file has to be treated differently. Every time the server application starts and closes, appropriate entries with the date and time are written logged and every logged item also includes a timestamp.

The logging functionality includes extensive information for each action taken on the server's side and details for each packet received, the way that it was retrieved and its payload. Generally, the log feature is implemented in a way so that it enables an offline studies of what precisely occurred during a search and rescue mission. This is important for several reasons. Firstly, the mountain rescue team can playback their missions and analyse how efficient their performance was, highlighting anything that went wrong or indeed anything that went right. This playback can also be used as evidence in legal hearings, such as a coroner's inquest into a fatality. Another important advantage is that an analysis can be made of the chosen connectivity option that each client used in various locations within the search and rescue area. Such studies can facilitate improved network support in the area and help anticipate connectivity problems.

5.5. Hardware

The hardware requirements of the server component are much simpler to satisfy than the client as the server is located at a fixed location, has no limited resources and only needs to be able to listen to and process all the messages that the clients send.

Therefore, a regular PC with a suitable Internet connection can be used as a server with a network interface listening to packets sent from the clients either via WiFi or GPRS. Since there is no large quantity of data being sent to the server, the Internet connection does not need significant data rates and a normal broadband connection will suffice.

The decision whether a PC should have one network interface listening for both IPv4 and IPv6 packets (using two sockets) or two different network interfaces listening individually for IPv4 and IPv6 packets is merely an implementation issue. However, it is imperative for the server to have a global address (IPv4 and/or IPv6) so that it is reachable by clients from anywhere in the Internet.

Apart from the IP based connectivity options that were defined for the LAS, a client also has the ability to send GPS coordinates via SMS and thus the server should have a GSM modem to receive these packets. Generally, there are various GSM modules (some of them depicted in Figure 17) that could be used to provide the SMS service to the server application. We have successfully used a standard Nokia 6230i mobile phone using a Bluetooth connection between the Nokia 6230i and the server with the aid of a Widcom USB-Bluetooth dongle on the server's side.

GSM Module 9/18 ([62]) *Falcom C2D-SI-GPRS ([63])* *Nokia 12i ([64])*

Figure 17 GSM Modules

Figure 18 Maestro 100 GSM/GPRS Modem

However, we have now settled with a Maestro 100 GSM/GPRS modem (Figure 18). This modem is quad-band, connects to the server via RS-232 and can be used with an external antenna to boost receiver gain in areas with low GSM signal.

5.6. Software

The application on the server side could be theoretically developed with almost any programming language suitable to handle network sockets and the connection with the GSM modem. For reasons of simplicity and a comprehensive development environment, it was decided to use the .NET framework v2.0 and Visual C# for the development of the server application.

Using Visual C# has the disadvantage that the code would need to be ported to C/C++ or Java to run on another platform (e.g. BSD or Linux). This would require a significant effort, albeit entirely possible. On the other hand, it is more important for the end users (in this case members of a mountain rescue team) to have as much familiarity with a new system as possible. Since the server application runs under the extremely popular Microsoft Windows XP or Vista, the vast majority of users will only have to be trained to use the application itself rather than also having to be trained in basic use of the underlying system.

5.7. Alerting

The LAS has now been successfully integrated with AlarmTILT and to combine a general Alerting system with Presence Management. This integration will be described in detail in a future deliverable, D3.3.1 [3], but we provide a high-level description of the integration here.

The HQ of the mountain rescue team serves as the control room for an emergency scenario in addition to being a rendezvous point for rescue workers responding to an emergency call. Currently, the HQ is notified by the Police when there is an emergency call and the relevant information is then sent over a paging system to all rescue volunteers. Once a suitable number of volunteers arrive at the HQ, they are briefed and then despatched to the appropriate search locations.

Rescue volunteers have a portable device (currently the HP iPAQ 6915) which automatically sends periodic location updates using the LAS client. The portable devices that the rescue workers have can be ‘activated’ automatically as soon as an emergency call occurs and only ‘de-activated’ once the search and rescue mission has been completed.

Using an alert system, the client application is activated automatically when both the following conditions are true:

1. An emergency notification is received
2. The rescue worker is able to respond to the emergency

The AlarmTilt system from M-PLIFY is being used as the alert system. using a modified version of the AlarmTilt SOAP API we have implemented a system that alerts rescue workers to emergency calls and replaces (or is complimentary to) the current paging system. Messages can be sent via email, SMS, voicemail or a bespoke client-server messaging system.

5.7.1. Integration Example

A general overview of an example follows:

1. An emergency call is made to the Police who then contact the Mountain Rescue HQ and inform the controller of the details of the emergency.
2. The controller enters the details of the emergency into the AlarmTILT client component and 'activates' the alarm. The details of the emergency are written to the operations database, located in the HQ. The operations database records all relevant information that happens in the course of a search and rescue operation. This information is used to generate logs and statistics for later analysis by the Mountain Rescue Team.
3. The AlarmTILT client component checks the Mountain Rescue Team's personnel database (located at the HQ) and sends alerts (via the AlarmTilt server) to all personnel registered as being available for this date and time. The precise nature of these alerts per rescue worker can be configured accordingly in the database.
4. Each rescue worker receives the alert on his/her device and accepts or declines to respond to the emergency call. A reply of 'accept' will automatically trigger the LAS client to start sending location updates to the LAS server at the HQ.
5. The AlarmTILT component collates all the replies by polling the AlarmTilt server and enters the details into the operations database accordingly.

Once all the replies have been collated and entered into the operations database, the AlarmTILT client component has done its job. The continuing monitoring and management of the search and rescue operation is conducted by a combination of the LAS server, the databases and the general mountain rescue backend application. Any events during the course of the search and rescue operation are handled in a similar fashion.

6. Evaluation/Findings from Tests

The main scope of the LAS is to provide location awareness to the mission coordinator and enable him to take more informed decisions and organize more efficiently the missions of the team in a mountain rescue domain. In our effort to facilitate the concept of location awareness we have envisaged, designed and implemented the previously presented system. Its evaluation process has been separated into two phases, namely the lab/campus phase and the mountain phase based on the place where experiments were conducted.

6.1. Campus Tests

The campus evaluation phase included the server application running in our lab and a people roaming around the Lancaster University campus with the HP iPAQ 6915 device running the LAS client application and transmitting GPS coordinates to the server. Around the campus the LAS client is able to use either the WiFi network or a GPRS connection. In addition, when neither were available, or when the LAS client was trying to establish a GPRS connection, the SMS option was successfully used. Figure 19 shows the WiFi coverage around the Lancaster University campus and Figure 20 shows the different connectivity options used during one test roaming around the campus.

Figure 19 Lancaster University Campus WiFi Coverage

Results from this set of experiments show that the application successfully supported the concept of location awareness and the person watching the server application running was able to identify the precise location of the person roaming around campus (something that was also validated via mobile phone calls or by using PMRs). It was during this testing that the hardware and software restrictions previously mentioned were identified. Windows Mobile 5.0 cannot have two IP-based connections active at the same time and be able to hot swap from one to the other. Our intention was to be able to utilize the WiFi connectivity option and have on stand-by the GPRS connection, which takes approximately 10-20 seconds to be set up, to take over when the WiFi connectivity was lost. However, Windows Mobile 5.0

cannot handle clearly and efficiently two concurrent IP connections and thus this creates an overhead when trying to establish a GPRS connection. In addition, Windows Mobile 5.0 has a bug when trying to swap from GPRS to the WiFi network, and sometimes does not inform the client application correctly about the presence of the WiFi network. This bug was examined thoroughly and it was identified that even when the LAS client application was shutdown the PDA could not be connected to the WiFi network. Figure 20 graphically represents the location of the retrieved GPS coordinates and the connectivity option used for their transmission during one experiment on the Lancaster University campus. It was expected that the LAS client would utilise the WiFi connectivity option a lot more during the experiment as there are specific areas along the route that offer (based on Figure 19), good WiFi coverage. By carefully examining the log file of the client it was identified that the LAS client did not get any indication from the Operating System that the device could connect to eduroam[1]. Further analysis revealed that when the device had established a GPRS connection, the Operating System could not connect to the eduroam WiFi network from access points along campus even when if the LAS client application was shutdown. Furthermore, the device was unable to connect even when the wireless adapter was restarted

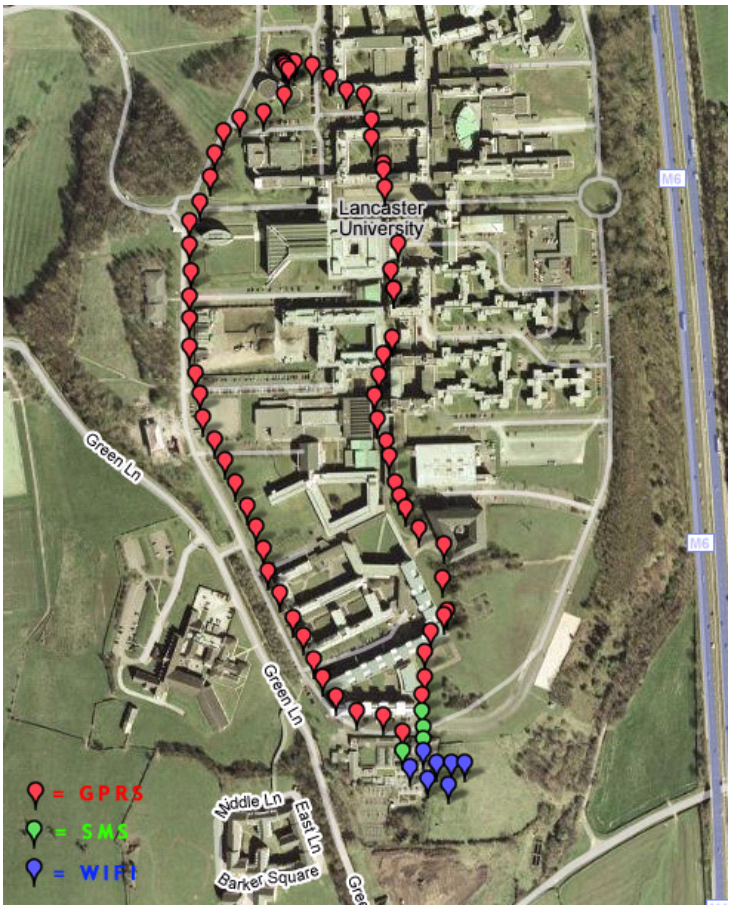

Figure 20 Connectivity Options Used During One Test on Campus

[1] eduroam is an SSID for WiFi networks which allows users of academic institutions to connect to the WiFi networks of other academic institutions. See [28] for more details.

After thorough investigation identified that there are many similar issues reported for PDAs that run Windows Mobile unable to connect to WiFi after having initialized a GPRS connection. In addition, correspondence with Peter Foot, a Microsoft Device Application Developer, revealed that *"having GPRS and WiFi connections concurrently is not a well supported scenario"* in devices running Windows Mobile 5.0 or 6.0 [29].

Another interesting point in some of the results of the campus experiments is that the LAS server sometimes never receives messages that were sent by the LAS client.

In one example, the client application sent 9 packets over WiFi but the server managed to receive only 7 of them. By carefully examining the log of the server it was identified that the packets with sequence numbers 00005 and 00006 did not arrive. The log snippet from the client that is presented in Figure 21 identifies that these packets were sent when the device was exiting InfoLab21. As can be noticed in the log the WiFi signal indication is really poor (-90db) and therefore the application identifies that it should swap from WiFi to GPRS, according to the Flow Chart presented in Figure 11. According to the implementation of this procedure our application instructs the Connection Manager to swap from WiFi to GPRS but as it can be noticed from the log that the device continues transmitting packets over the WiFi network. This behaviour was discussed in Section 4.8.1 (Software Restrictions) as, unfortunately, it is known that the Connection Manager operating component does not always follow the application's connection request.

```
[13:34:19] Found eduroam, IP:148.88.249.252, Signal:-90
[13:34:21] Found eduroam, IP:148.88.249.252, Signal:-90
[13:34:22] Timer Ticked - Sending Coordinates
[13:34:22] Current signal : -90
[13:34:22] Current way : WIFI
[13:34:22] I will send coordinates with : GPRS
[13:34:22] Current way is different from the one that i want to
use
[13:34:22] Changing networks
[13:34:34] I will send this : #WIFI#101#1234#00005#13:34:34:3434
04/09/07#54@00.3187"N#002@47.0920"W#normal#
[13:34:34] Sending packet via WIFI
[13:34:34] WIFI pcks : 5
[13:34:35] Found eduroam, IP:148.88.249.252, Signal:-90
[13:34:35] Found eduroam, IP:148.88.249.252, Signal:-90
[13:34:37] Timer Ticked - Sending Coordinates
[13:34:37] Current signal : -90
[13:34:37] Current way : WIFI
[13:34:37] I will send coordinates with : GPRS
[13:34:37] Current way is different from the one that i want to
use
[13:34:37] Changing networks
[13:34:37] I will send this : #WIFI#101#1234#00006#13:34:37:3437
04/09/07#54@00.3187"N#002@47.0920"W#normal#
[13:34:37] Sending packet via WIFI
[13:34:37] WIFI pcks : 6
[13:34:37] Found eduroam, IP:148.88.249.252, Signal:-90
[13:34:39] Found eduroam, IP:148.88.249.252, Signal:-90
```

Figure 21 Log snippet from one of the campus tests

However, outside of these bugs the client application proved to be working correctly using the best-suited connectivity option that the Operating System was reporting as available. Results also prove that the

reordering and authentication mechanisms on the server side were working correctly and properly filtering messages.

6.2. Mountain Tests

Entering the mountain testing phase, we have evaluated our system in the actual region that the CMRT operates in by emulating search patterns of rescue missions. In all of our mountain tests we disabled the GPRS functionality on the PDA devices for two reasons: 1) to avoid the GPRS to WiFi swapping bug in Windows Mobile and 2) because GSM signals in the region are generally not good enough to establish or maintain GPRS connections.

In a preliminary test before we had established our on-mountain WiFi network, we conducted a walk and drive around the areas of Buttermere, Scale Bridge, Scale Hill and Lorton. During this 125 minutes trial the client application used both online and offline mode and sent 327 SMS messages with GPS updates to the server application. A person monitoring the server application was able to track our movement and identify that we were sending more frequent GPS messages when driving rather than when we were walking. An interesting point to mention is that the server application received 293 SMS messages, meaning that 24 messages did not arrive at the server application. By carefully examining the log we have identified that when the client application was sending SMS messages more frequently, the GSM network had higher chances of not delivering these messages.

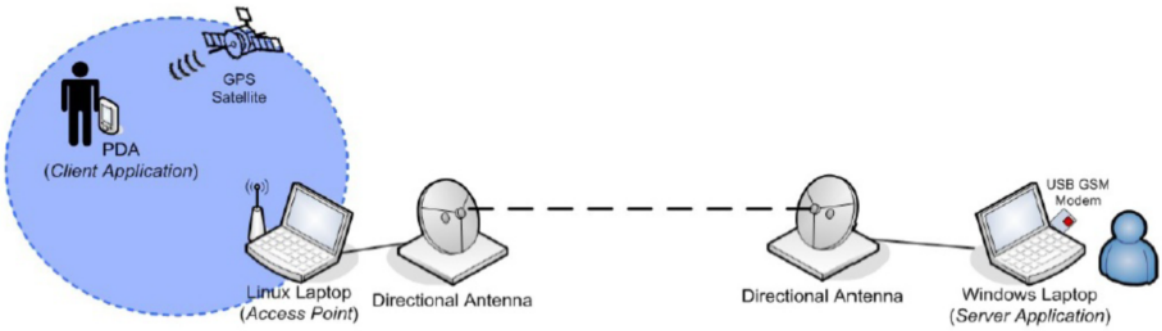

Figure 22 On-mountain testbed at Buttermere

A further on-mountain experiment was carried out using an on-mountain WiFi network testbed for about one hour in the area of Buttermere (Cumbria, UK). During this experiment, a person was monitoring the LAS server application running on a laptop tracking the movements of another person who was roaming in a radius of 500m around the access point, holding the PDA with the LAS client application (see Figure 22). Table 6 presents results from the packets being transmitted either in the form of IP packets over WiFi, or SMS messages over the GSM in online or offline manner.

Table 6 Results from one on-mountain test

	Client (packets sent)		**Server** (packets received)	
Connectivity Option	Online	Offline	Online	Offline
WiFi	121	18	114	18
SMS	42	9	40	9

Regarding the packets being transmitted over WiFi using TCP, it has been observed that only 7 out of 121 (approximately 5.8%) were lost, which is not a poor percentage for a wireless network taking into consideration the physical characteristics of the environment and the mobility of the person holding the client device. Furthermore, it has been noticed that when the client application was identifying a broken TCP socket it was swapping to GSM, managing to minimize the packet loss.

Regarding the SMS messages, it has been observed that two of them were not received from the server application, a behaviour that is attributed to the GSM provider. On the server side, the application was able to rearrange out of sequence packets and split merged TCP packets, presenting on the map the movement of the person roaming.

More recent tests have been conducted integrated with the deployment of our on-mountain network solution, similar to the right-hand side of the configuration shown in Figure 1, establishing uplink connections to CLEO and GEANT (via the Astra2Connect satellite service), with the LAS server located at the Cockermouth HQ. Our current set of tests show that the client's complex mechanism for identifying the availability and utilizing the connectivity options works relatively well by basing its primary functionality on two IP-based connectivity options and sending SMS messages as a backup solution in order to increase the redundancy for the system.

Generally, it can be determined that the LAS client application can transmit GPS coordinates successfully with respect to the available connectivity options. Furthermore, the person monitoring the server application can accurately track the movements of the people holding the client PDAs and although further tests have to be carried out, it can be stated that the developed LAS manages to provide the concept of location awareness in a mountainous domain.

7. Location Awareness using Location Based Services

To complement the LAS using GPS, we have also conducted an empirical study of the accuracy of GSM Location Based Services (LBS) in densely and sparsely populated areas of the UK. The motivation for this was twofold:

1. to see if LBS was accurate enough to be used to monitor the locations of rescue workers
2. to determine if LBS was useful for locating lost subjects

We used real-world data of three network providers (O2, Orange and Vodafone) to compare locations reported by their LBS (Location Based Service) with locations gained from GPS.

In many of their rescue missions, the mountain rescue team are in telephone contact with the casualty via their mobile phone. In these situations, it is obvious that a positioning technology that could accurately locate the casualty and supply that vital information to the search and rescue coordinator would be invaluable. In addition, should rescue workers not have a sufficient GPS signal to determine their location, they may still have a connection to a GSM network (via their PDA) that offers LBS. Thus, it is theoretically possible for the mission coordinator to determine the locations of rescue workers based primarily on their GSM connection. However, the accuracy of the returned LBS information is open to question.

Our requirements were therefore based on the need to determine the ‘‘real” accuracy of LBS technologies not only at one instance in time, but periodically, to determine how much emphasis we can place on the results supplied to us by the network operators.

The overall objective of the study was to provide a comprehensive view of accuracy in cellular positioning technology currently deployed by the three most popular operators in the UK: O2, Orange and Vodafone. In particular, its objectives were to evaluate:

- the accuracy as claimed by operators compared to “real” accuracy, as actually achieved in both densely and sparsely populated areas the impact of base station location on claimed and real accuracy
- the ability to infer the respective positioning technology based on any or all of the data aggregated during this study

The data collection was completed using a mobile phone and a GPS-enabled PDA running a purpose-built piece of software, thereby providing the means for continuously assessing the operators’ progress in improving upon positioning accuracy. Positioning data for dozens of LBS requests were collected in both a densely and sparsely populated area of the UK. The specific locations were Ambleside in the Lake District, with 32–64 inhabitants per square kilometre, and the city of Manchester, with more than 1000 inhabitants per square kilometre, hence two very distinctive locations in terms of population. Figure 23 and Figure 24 contrast the two types of environments in terms of base station distribution to highlight the differences.

To that effect, a mobile phone was located at various locations, and reference locations in the form of GPS coordinates were recorded for each provider, in both a sparsely and a densely populated area. In effect, a dataset was created for each location, which consists of the position and accuracy as indicated by the corresponding provider, a reference location and upon data aggregation, the position of the nearest provider-operated base station.

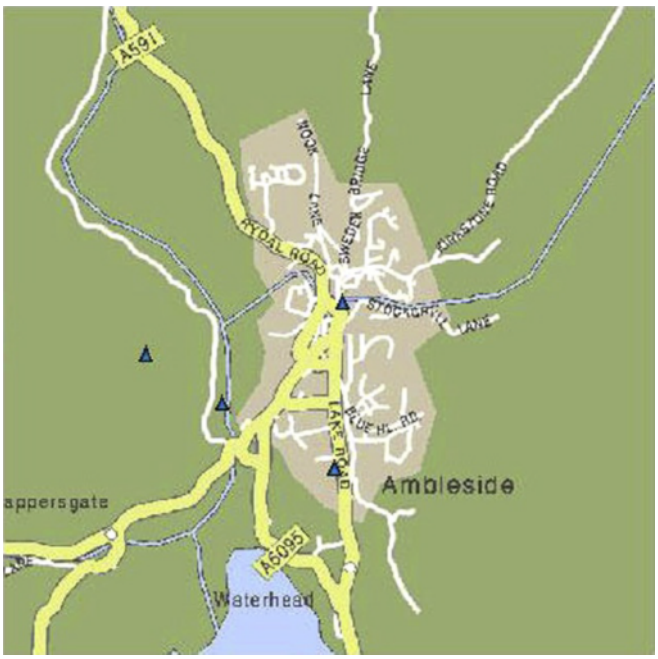

Figure 23 Base Stations in Ambleside

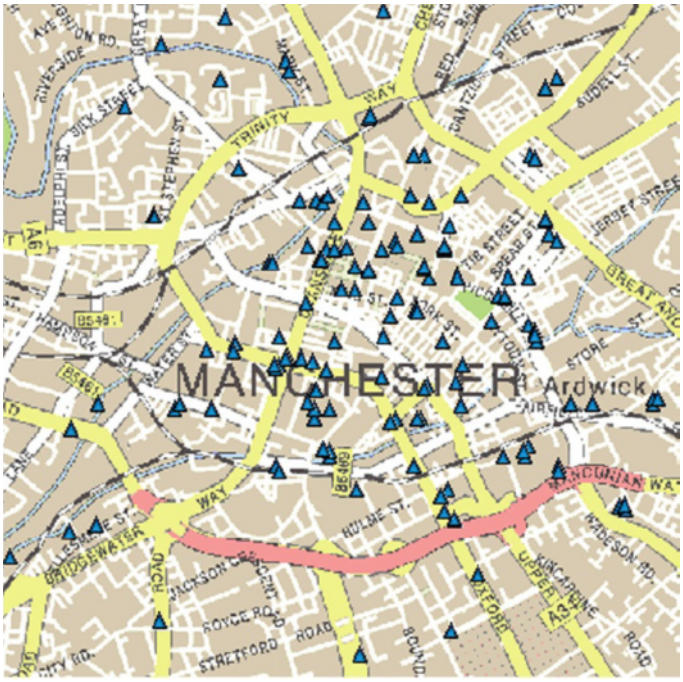

Figure 24 Base Stations in Manchester

So as to make the data collection process as objective as possible, a grid of destined collection points was laid on top of each environment. Each grid was chosen to be 200m^2, the vertices of which constitute the data points. In an attempt to minimise preference of any network, we overlaid this grid such that the average distance to the closest base station and closest high-elevation (higher than 10m) base station of each net work was comparable and without influence on our forthcoming analysis.

A total of 21 data points were collected in Ambleside and 33 in Manchester for each network, except for O2 in Manchester, where only 32 points were collected. It should be noted that the mobile phone was not in roaming mode at any point in time, preventing potential falsification of network-based positioning information.

The components used to obtain the necessary data were selected as follows:

- *Reference positioning information*: Pretec BluetoothGPS mini, a SiRF II (12 parallel channels) GPS receiver supporting the NMEA-0183 standard (v. 2.20) and the satellite-based augmentation systems WAAS (North America) and EGNOS (Europe).
- *Mobile phone positioning*: World-Tracker, which acts as a Middleware service that allows individual end users to interface with the LBS component of multiple mobile network operators, was used to perform cellular positioning for all networks included in this study.
- *Mobile phone*: HP iPAQ h6340.
- *Base station information*: Retrieved from Ofcom's Sitefinder service [30].
- *Population density*: Areas selected based on the Center for International Earth Science Information Network's (CIESIN) Global Rural-Urban Mapping Project (GRUMP) population density data of the UK at a 30 arcsec resolution [31].

The data collected during this experiment indicated that a direct correlation between population density and both claimed and actual accuracy exists. In Manchester, we found that the overall accuracy across all providers is approximately 266m with O2's overly positive average of 112 m considerably improving the overall results. In Ambleside, we discovered an eightfold decrease in accuracy in idle-mode positioning and a 60% decrease in the case of Vodafone's dedicated-mode positioning.

Regarding the mountain rescue domain, we concluded that the currently deployed LBS techniques are not adequate to provide sufficiently accurate location information. None of the operators' accuracy claims were insufficient to serve modern applications, unless a rough indication as to the whereabouts of a mobile phone or PDA is satisfactory. For example, in the case of the mountain rescue domain, this level of result means that it could at most serve as supplemental aid, rather than a primary tool. The quality of a position measurement is only ever as good as the quality of its confidence interval. Thus, in rural areas where the confidence interval is low, the accuracy of the positioning information is also low. As a tool for monitoring the positions of rescue workers, LBS is next to useless when we consider that the general location of the rescue workers will already be know. However, LBS may have some use in locating a casualty when even the general location of the individual is not known. For example, even locating a casualty with an error of ± several kilometres can help narrow down the search area for the rescue workers.

8. Requirements Verification

According to the requirements defined in Table 1 of Chapter 2, the LAS implementation satisfies almost all of the defined requirements for the presence management solution. The only requirement that fails to be met is R09, which requires a battery runtime of 4 hours or more. Unfortunately, the iPAQs we tested could not achieve more than 2 hours battery life at full usage. However, this is not attributable to the LAS itself (although ways of conserving power consumption could be investigated) and with the general trend towards more energy-efficient devices and powerful batteries this problem may not exist in the short to medium term.

A summary of how well the presence management requirements are met by the LAS is provided in Table 7.

Table 7 Summary of the Requirements Verfication

Requirements for Presence Management		
R01	The client device must be able to acquire GPS coordinates in real-time.	✓
R02	The client must identify which connectivity options are available.	✓
R03	The client must utilise the best-suited connectivity option and seamlessly swap between the connectivity options based on pre-defined criteria.	✓
R04	Both the server and client must support both IPv4 and IPv6	o
R05	The client must transmit GPS coordinates in user-defined intervals.	✓
R06	The client device should be small and lightweight.	✓
R07	The client device should be wearable.	o
R08	The client device should be waterproof.	o
R09	The client device should have a battery runtime of at least 4 hours.	×
R10	The server must be able to listen for GPS coordinates regardless of the connectivity option that the clients use to send them.	✓
R11	The server must identify the exact node that sent a message and the exact connectivity option that was used.	✓
R12	The server must identify and reorganise messages received in the wrong order.	✓
R13	The server must display the locations of the client devices on suitable map displays.	✓
R14	The server must log all movements made by the clients for later analysis.	✓
R15	The client software should not need to be configured by rescue workers before use.	✓
R16	The client GUI must be simple and utilise large buttons for use wearing gloves.	o

✓	Meets the requirement
o	Partially meets the requirement or needs further work
×	Fails to meet the requirement

Abbreviations

CLEO	Cumbria and Lancashire Education Online
EDGE	Enhanced Data Rates for GSM Evolution
EGPRS	Enhanced GPRS (also known as EDGE)
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile communications
GUI	Graphical User Interface
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
JANET	Joint Academic Network
LAS	Location Awareness System
LoS	Line of Sight
MAN	Metropolitan Area Network
MANET	Mobile Ad-hoc Network
MANEMO	MANET and NEMO
MIPv6	Mobile IPv6
NEMO	Network Mobility
NMEA	National Marine Electronics Association
PDA	Personal Digital Assistant
SDK	Software Development Kit
UMF	Unified Message Format
UMTS	Universal Mobile Telecommunications System
WiFi	Wireless Fidelity

References

[1] U-2010 Deliverable 1.1.1, "Reference scenarios based on user studies", October 2007.

[2] U-2010 Deliverable 1.1.2, "Functional requirements for networks and services", March 2007.

[3] U-2010 Deliverable 3.3.1 "Report on Interworking Solutions", Work in Progress.

[4] U-2010 Deliverable 4.2.1, "Report on the Mountain Rescue Service Concept", July 2008.

[5] U-2010 Deliverable 4.2.2 "Prototype Mountain Rescue Team Service Trial", Work in Progress.

[6] U-2010 Deliverable 4.3.1, "Concept of Fire Services Solution", July 2008.

[7] M. Mohr, C. Edwards, B. McCarthy, "A Study of LBS Accuracy in the UK and a Novel Approach to Inferring the Positioning Technology Employed", Computer Communications (2008), doi:10.1016/j.comcom.2008.01.039.

[8] Cockermouth Mountain Rescue FAQ, available from: http://www.cockermouthmrt.org.uk/faq.aspx#faq4

[9] Lancaster University Network Mobility Group homepage, http://www.network-mobility.org/

[10] G. Baddeley. "GPS – NMEA sentence information", January 2001, available from: http://aprs.gids.nl/nmea/#rmc

[11] OpenNETC Consulting, LLC. "Compact Framework". Available from http://www.opennetcf.com/CompactFramework/tabid/85/Default.aspx

[12] G. Baddeley. "Glenn Baddley GPS – NMEA sentence information". 2007. Available from: http://home.mira.net/~gnb/gps/nmea.html

[13] G. Baddeley. "GPS – NMEA sentence information". January 2001. Available at: http://aprs.gids.nl/nmea/#rmc

[14] Google Corporation. "Google Maps API Concepts". Available from: http://www.google.com/apis/maps/documentation/index.html

[15] Microsoft Corporation. "Windows CE Networking Team Weblog". Available from: http://blogs.msdn.com/cenet/archive/2006/06/06/620360.aspx

[16] Microsoft Corporation. "Connection Manager". 2007. Available from : http://msdn2.microsoft.com/en-us/library/bb416435.aspx

[17] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", IETF Request for Comments: 4380, February 2006.

[18] HP Corporation. "Additional Product Information – Ipaq 6915". Available from http://h20000.www2.hp.com/bc/docs/support/SupportManual/c00713094/c00713094.pdf

[19] Sun Corporation. "Java ME Technology". Available from: http://java.sun.com/javame/technology/index.jsp

[20] D. Fitton, "Java Support on Pocket PC". Available from: http://www.comp.lancs.ac.uk/~fittond/ppcjava.html

[21] Nokia Corporation."Nokia tools and Nokia SDKs for developers". Available from: http://www.forum.nokia.com/main/resources/tools_and_sdks/listings/index.html

[22] Sun Corporation. "Java ME – Java Technology for the Wireless Industry (JTWI), JSR185". Available from: http://java.sun.com/products/jtwi/

[23] S. Berka. "Java for PocketPC PDA's". Available from: http://www.berka.name/stan/jvm-ppc/java_for_pda.html

[24] Microsoft Corporation. ".NET Framework". Available from : http://msdn2.microsoft.com/en-gb/netframework/default.aspx

[25] Microsoft Corporation. ".NET Compact Framework". Available from: http://msdn2.microsoft.com/en-gb/netframework/Aa497273.aspx

[26] S. Mayr. "Sending short messages (SMS) via GSM mobile phones". Available from: http://www.scampers.org/steve/sms/

[27] Lancaster University Information System Services WiNET, The Campus Wireless Network Coverage Map. Available from : http://www.lancs.ac.uk/iss/winet/coverage-map.htm

[28] eduroam homepage. Available from http://www.eduroam.org/

[29] P. Foot. "Windows Mobile 6.0 and Connection manager". Micorosft's Forum. Available from: http://forums.microsoft.com/MSDN/ShowPost.aspx?PostID=2088878&SiteID=1&mode=1

[30] Ofcom Sitefinder, Mobile Phone Base Station Database http://www.sitefinder.ofcom.org.uk/

[31] Center for International Earth Science Information Network (CIESIN), Columbia University; International Food Policy Research Institute (IFPRI); The World Bank; and Centro Internacional de Agricultura Tropical (CIAT). Global Rural-Urban Mapping Project (GRUMP), Alpha Version: Population Density Grids. Palisades, NY: Socioeconomic Data and Applications Center (SEDAC), Columbia University, 2004. Available from: http://sedac.ciesin.columbia.edu/gpw