

**INTEGRATING MOBILE DEVICES INTO GRID  
APPLICATIONS**

**MSc. Thesis by  
Ersan ÖZTÜRK**

**Department : Computer Engineering**

**Programme: Computer Engineering**

**Supervisor : Asst. Prof. Dr. D. Turgay Altılar**

**JUNE 2007**

**INTEGRATING MOBILE DEVICES INTO GRID  
APPLICATIONS**

**M.Sc. Thesis by  
Ersan Öztürk**

**(504041515)**

**Date of submission : 7 May 2007**

**Date of defence examination: 11 June 2007**

**Supervisor (Chairman): Asst. Prof. Dr. D. Turgay ALTILAR**

**Members of the Examining Committee: Prof. Dr. Şebnem BAYDERE**

**Assoc. Prof. Dr. Sema OKTUĞ**

**JUNE 2007**

**GRID UYGULAMARINA MOBİL CİHAZLARIN  
ENTEGRASYONU**

**YÜKSEK LİSANS TEZİ**

**Ersan ÖZTÜRK**

**(504041515)**

**Tezin Enstitüye Verildiği Tarih: 7 Mayıs 2007**

**Tezin Savunulduğu Tarih: 11 Haziran 2007**

**Tez Danışmanı : Yrd. Doç. Dr. D. Turgay ALTILAR**

**Diğer Jüri Üyeleri: Prof. Dr. Şebnem BAYDERE**

**Doç. Dr. Sema OKTUĞ**

**HAZİRAN 2007**

## CONTENTS

<b>ABBREVIATIONS</b>	<b>v</b>
<b>LIST OF TABLES</b>	<b>vi</b>
<b>LIST OF FIGURES</b>	<b>vii</b>
<b>SUMMARY</b>	<b>viii</b>
<b>ÖZET</b>	<b>ix</b>
<b>1. INTRODUCTION</b>	<b>1</b>
1.1. Problem Definition	1
1.2. Current Studies and Applications	2
1.3. The Approach	2
<b>2. BACKGROUND AND MOTIVATION</b>	<b>4</b>
2.1. Grid Computing And Wireless Devices	4
2.1.1. Grid computing	4
2.1.2. Integrating wireless devices into grid	8
2.1.3. Applications	13
2.2. Wireless Devices	14
2.2.1. Wireless devices and their limitations	14
2.2.2. Wireless communication alternatives	16
2.3. Global Positioning System	17
2.3.1. Applications	19
2.3.2. Future of global positioning system	20
2.4. Motivation	20
<b>3. THE IMOGA AND CIM-TR</b>	<b>22</b>
<b>4. IMPLEMENTATION OF CIM-TR</b>	<b>28</b>
4.1. Client Application	28
4.1.1. Decision making on XML parameters	28
4.1.2. Message structure	31
4.1.3. User interface	33
4.2. Server Application	34
4.2.1. GDF	35
4.2.2. Road database	37
	iii

4.2.3. Locating algorithms	39
4.3. Communication	40
4.4. Traffic calculation and display application	44
4.4.1. Who is online	46
4.4.2. Traffic info	48
<b>5. EXPERIMENTAL RESULTS</b>	<b>51</b>
5.1. Experiment 1	51
5.2. Experiment 2	53
5.3. Experiment 3	54
<b>6. CONCLUSION</b>	<b>56</b>
<b>7. FUTUREWORK</b>	<b>57</b>
<b>REFERENCES</b>	<b>58</b>
<b>APPENDIX</b>	<b>61</b>
<b>RESUME</b>	<b>64</b>

## ABBREVIATIONS

<b>GPS</b>	: Global Positioning System
<b>XML</b>	: Extended Mark-up Language
<b>GPRS</b>	: General Packet Radio Service
<b>GDF</b>	: Geographic Data Files
<b>WAP</b>	: Wireless Application Protocol
<b>I-WAY</b>	: Information Wide Area Year
<b>NFS</b>	: Number Field Sieve
<b>OGSA</b>	: Open Grid Service Architecture
<b>VO</b>	: Virtual Organization
<b>DNS</b>	: Domain Name Server
<b>API</b>	: Application Programming Interface
<b>SDK</b>	: Software Development Kit
<b>P2P</b>	: Peer to Peer
<b>SOAP</b>	: Simple Object Access Protocol
<b>WSDL</b>	: Web Services Description Language
<b>PDA</b>	: Personal Digital Assistant
<b>QoS</b>	: Quality of Service
<b>GSI</b>	: Grid Security Infrastructure
<b>UDDI</b>	: Universal Description, Discovery and Integration
<b>GSS</b>	: Generic Security Service
<b>MDS</b>	: Monitoring and Discovery Service
<b>COA</b>	: Care Of Address
<b>CLDC</b>	: Connected Limited Device Configuration
<b>CDC</b>	: Connected Device Configuration
<b>MIDP</b>	: Mobile Information Device Profile
<b>IMEI</b>	: International Mobile Equipment Identity
<b>GSM</b>	: Global System for Mobile communication
<b>MSC</b>	: Mobile Switching Centre
<b>PRC</b>	: Pseudo Random Code
<b>MCS</b>	: Master Control Station
<b>FGDC</b>	: Federal Geographic Data Committee
<b>ISP</b>	: Internet service Provider

## LIST OF TABLES

	<u>Page No</u>
<b>Table 2.1</b> : Bandwidth and ranges for wireless technologies.....	16
<b>Table 4.1</b> : Results of client versions in experiment 1.....	29
<b>Table 4.2</b> : Results of client versions in experiment 2.....	31
<b>Table 4.3</b> : Location table in the database.....	34
<b>Table 4.4</b> : Main tables from database for Level 0.....	37
<b>Table 4.5</b> : Main tables from database for Level 1.....	37
<b>Table 4.6</b> : Complex_Features table from database for Level 2.....	37
<b>Table 4.7</b> : User_last_location and Road_speed tables.....	38
<b>Table 4.8</b> : Actions to take in TwoCandidateRoads.....	43
<b>Table 4.9</b> : Percentage contribution of sets to the result.....	49
<b>Table 4.10</b> : Display colors for the road traffic.....	49
<b>Table 5.1</b> : Number of locations handled by Server application in 24 hours.....	53
<b>Table 5.2</b> : Success rate of road locating procedure.....	54
<b>Table 5.3</b> : Displayed results according to data sets .....	55

## LIST OF FIGURES

	<u>Page No</u>
<b>Figure 2.1</b> : Grid Protocol Architecture.....	7
<b>Figure 2.2</b> : QoS specification XML message.....	13
<b>Figure 2.3</b> : GPS Triangulation.....	18
<b>Figure 3.1</b> : IMOGA architecture.....	22
<b>Figure 3.2</b> : CIM-TR architecture.....	24
<b>Figure 3.3</b> : Uploaded Products by Interlocutors.....	25
<b>Figure 3.4</b> : Sample Grid System .....	26
<b>Figure 4.1</b> : The user interface of 3 <sup>rd</sup> client version in 2 <sup>nd</sup> experiment.....	30
<b>Figure 4.2</b> : The fields in the header.....	31
<b>Figure 4.3</b> : The fields in the location_data.....	32
<b>Figure 4.4</b> : User Interface of client application.....	33
<b>Figure 4.5</b> : GDF, Level 0, Topology: The fields in the header.....	35
<b>Figure 4.6</b> : GDF Level 1, simple features.....	36
<b>Figure 4.7</b> : GDF Level 2, complex features.....	36
<b>Figure 4.8</b> : IfOnRoad algorithm.....	39
<b>Figure 4.9</b> : FirstLocationInfo algorithm.....	40
<b>Figure 4.10</b> : IfOnTheLineWithDirection algorithm.....	41
<b>Figure 4.11</b> : FirstLocationInfo and IfOnTheLine Procedures Illustration.....	42
<b>Figure 4.12</b> : TwoCandidateRoads algorithm.....	42
<b>Figure 4.13</b> : Istanbul map is divided into $5 \times 5 = 25$ equal components.....	46
<b>Figure 4.14</b> : World Coordinate System vs Java Coordinate System.....	47
<b>Figure 4.15</b> : Part of Who is online application screen.....	48
<b>Figure 4.16</b> : Display application interface.....	50



## **INTEGRATING MOBILE DEVICES INTO GRID APPLICATIONS**

### **SUMMARY**

Integrating mobile devices into Grid technologies and server applications can give ability to command power of supercomputers with a mobile device on one hand and can allow big applications to reach important data anywhere, anytime, on the other. This project is planned to be an example to gather and share data that can be collected by ubiquitous mobile devices which can employ different kind of sensors such as GPS, temperature, health monitoring and pollution. In this project location and speed information that is produced by GPS enabled mobile devices such as mobile phones, is used. The developed client application running on mobile devices located in vehicles, such as the mobile phone of the driver, sends location and speed information to the server application in short time intervals via GPRS in the forms of XML like messages. The developed server application, which is preloaded with the highway coordinates via files in GDF format, locates the street that the vehicle is moving along and the received speed information is recorded along with a timestamp. A display application has also been implemented to calculate average of speeds at that very moment and post it on the Internet and WAP. If there is no actual data, i.e. there is no vehicle moving on a specific street, statistical data is utilized to produce such information. Thus foreseeing the traffic not only spatially but also in time is made possible. The client application running on the mobile device that feeds in data or any other devices that a client could name such as computers can exploit the information produced by the integrated system. Although producing and sharing traffic information is the essential aim of the project it is also shown that additional features such as vehicle tracking for even a systematic approach for traffic management can be done with this project.

## GRID UYGULAMARINA MOBİL CİHAZLARIN ENTEGRASYONU

### ÖZET

Mobil cihazların Grid teknolojilerine ve sunucu uygulamalarına entegrasyonu bir taraftan süper bilgisayarları bir mobil cihazla kumanda etmeye olanak sağlanırken diğer taraftan da büyük uygulamaların önemli verilere her yerde ve her zaman erişebilmesine olanak sağlayabilir. Bu çalışma, GPS, sıcaklık, sağlık izleme ve kirlilik gibi farklı çeşitteki algılayıcıları barındırabilecek aynı zamanda pek çok yerden veri toplamaya olanak sağlayacak mobil cihazlardan veri toplama, işleme ve paylaşma üzerine bir örnek olması üzere planlandı. Projede konum ve hız verisi üretebilecek GPS alıcısına sahip mobil cihazlara örnek olarak cep telefonları kullanıldı. Sürücünün cep telefonu gibi otomobil içerisinde yerleştirilen mobil cihazların üzerinde koşturmak üzere geliştirilen istemci, konum ve hız verisini kısa zaman aralıklarında XML mesajları formatında GPRS üzerinden sunucuya göndermektedir. GDF formatında ana yol koordinatları önceden girilmiş sunucu uygulaması, aracın üzerinde hareket ettiği yolu bulur, yol için hız verisini zaman damgası ile birlikte kaydeder. Hızların ortalamasını hesaplamak ve bunu Internet ve WAP üzerinden sunmak üzere bir de görüntüleme uygulaması geliştirilmiştir. Eğer güncel veri yoksa, yani eğer o anda o yol üzerinde veri aktaran bir araç yoksa, istatistiksel veri kullanılarak bilgi sunulur. Böylece trafiği sadece uzamsal değil ayrıca zamansal olarak önceden görmek mümkün olur. Mobil cihaz üzerinde koşturan istemci uygulaması veya herhangi bir bilgisayar entegre sistem tarafından üretilen bilgiyi kullanabilir. Her ne kadar projede geliştirilen uygulamanın asıl amacı trafik bilgisi üretmek ve paylaşmak olsa da araç takibi, hatta trafik yönetimi için sistematik yaklaşımlar bu proje tarafından mümkün kılınabilir.

## **1. INTRODUCTION**

In this project after giving the importance and benefits of integration of mobile devices into Grid applications, a sample architecture and application are presented.

### **1.1 Problem Definition**

Grid Computing is an emerging concept which changes conventional way of thinking in computer technology. Currently researchers on Grid Computing are concentrated on pooling the resources of high capacity processing and storage units with strong visualization centers to solve hard scientific problems [1]. We believe that integrating mobile devices into Grid technologies can enable a simple mobile device to command even supercomputers, and further more applications running on supercomputers can acquire data instantly regardless of geographic location and time. The only requirement is a wireless communication support.

According to Metcalfe's law as the number of devices and users increases, grid-based resources become more valuable [2]. Participating mobile devices can increase the number of users to a number which can not be compared to today's population in the Grid so that Grid based resources become more valuable.

We can get big profits by integrating wireless mobile devices into Grid applications but how such devices with limited capacities and features can join the Grid Systems? What kind of architecture can enable that? What are the features that these devices shall support to be in this architecture? Can mobile phones, which are spread world wide to almost any point human beings live, be used for this aim? What shall be the communication mechanism and structure? Even if mobile phones are developing very fast, what is the most valuable data we can get from mobile phones today? How can such data be processed and what kind of applications can be produced? In this project answers for these questions are given with our approach and a sample

application, which can collect, process and display data produced by the mobile devices, is introduced.

## **1.2 Current Studies and Applications**

There are some studies and applications developed using Wireless Grid even these are just far beyond the potential usage. Distributed Ad-Hoc Resource Coordination (DARC) lets devices with no prior knowledge of each other collectively record and mix an audio signal such as a concert, speech, lecture, or emergency event[3]. In the research carried out in Helsinki Institute of Physics called Gridblocks, healthcare application is developed that can detect the movement of lungs and heartbeats with 90% certainty within 50 cm distance of human body to produce valuable information of pulse and breathing rate[4]. The AKOGRIMO project is a European funded project aiming to architect and prototype Next Generation Grid based on Open Grid Services Architecture (OGSA) which exploits and closely co-operates with evolving Mobile Internet infrastructures based on IPv6[5]. The applications that are worked on or planned to be worked are e-Health, e-Learning and e-Crisis management. The GridLab project is one of the biggest European researches in the development of application tools and middleware for Grid environments. They are working on a project to develop theoretical scenarios and their implementation for using mobile devices to enhance the working environment and efficiency of application users[6].

## **1.3 The Approach**

In this project we suggest an architecture to address the issues of integrating mobile devices into a Grid system and sharing and producing information in a cooperative manner. The architecture is named as IMOGA (Integrating Mobile Devices into Grid Applications)

IMOGA architecture is tested against an application called CIM-TR(Cooperative Information Management for vehicle TRaffic flow) that collects data ubiquitously and produces information after processing this data. In CIM-TR mobile devices transfer their location and speed information to the server and server processes the received data and provides traffic information to data providers as well as the public.

## **2. BACKGROUND AND MOTIVATION**

Grid Computing is a fast growing research area in the last decade which has brought a new collective approach to problem solving. Integrating wireless mobile devices to Grid is a new concept which has also brought a new viewpoint to this research area. This chapter gives information about these researches, the state of the art in wireless devices today and GPS which is used in the application developed for this project. All these information is concluded with our motivation to make such a research.

### **2.1. Grid Computing and Wireless Devices**

The Grid Computing concept, evolution, architecture are given in details together with new research area of integrating wireless devices and applications below.

#### **2.1.1. Grid computing**

Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. The sharing is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource brokering strategies emerging in industry, science, and engineering. A set of individuals and institutions defined by such sharing rules form a *virtual organization (VO)*. VOs may vary tremendously in their purpose, scope, size, duration, structure, community, and sociology.

Computational grid is composed of hardware and software infrastructures that provide dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [7]. Resource sharing requires significant hardware infrastructure to achieve the necessary interconnections and software infrastructure to monitor and control the system. *Dependable* service is fundamental in Grid since users require assurances that they will receive predictable, sustained, and often high levels of performance from the diverse components that constitute the grid.

*Consistency* of service means standard services can be accessible via standard interfaces with standard parameters, like electric power grid. *Pervasive* access allows VOs to count on services always being available, within whatever environment they expect to move. Finally, if an infrastructure is wanted to be broadly accepted and used it must be offered with an *inexpensive* access.

When it is looked at the evolution of grid, there can be three different generations identified [1].

*First generation* started as projects to link supercomputing sites; which was called as metacomputing in that time. The origin of the term is believed to have been the CASA project which is one of several US Gigabit testbeds started in 1989. The objective of these early metacomputing projects was to provide computational resources to a range of high performance applications. FAFNER and I-WAY were the two representative projects of metacomputing. FAFNER was set up in 1995 to factor RSA130 using a new numerical technique called the Number Field Sieve (NFS) factoring method using computational web servers. It was capable of running on any workstation with more than 4 Mbytes of memory. I-WAY established in the same year and it was a means of unifying the resources of large US supercomputing centers. I-WAY was an experimental high performance network connecting many high performance computers and advanced visualization environments.

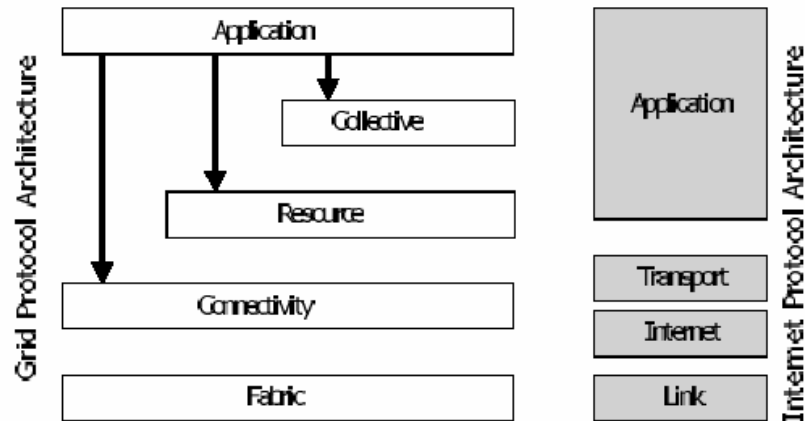
Second generation is the grid technology that set the standards and mostly used today. There are three important characteristics defined for grid system: Heterogeneity, scalability, adaptability where scalability and adaptability brings latency tolerance and dynamic behavior respectively to the grid. Middleware concept became the core component where it is defined as the layer of software sandwiched between the operating system and applications, providing a variety of services required by an application to function correctly. Middleware is used to hide the heterogeneous nature and provide users and applications with a homogeneous environment by providing a set of standardized interfaces to a variety of services. There are two important representatives of second generation Grid systems: Globus and Legion.

Globus provides a software infrastructure that enables applications to handle distributed heterogeneous computing resources as a single virtual machine. The Globus project is a U.S. multi-institutional research effort that seeks to enable the construction of computational grids started in 1997. A computational grid is a hardware and software infrastructure that provides dependable, consistent, and pervasive access to high-end computational capabilities, despite the geographical distribution of both resources and users. The evolution of Globus is continuing with the introduction of the Open Grid Service Architecture (OGSA) which is a Grid architecture based on Web Services and Globus. Legion on the other hand is an object oriented metasytem which has provided the software infrastructure so that a system of heterogeneous, geographically distributed, high performance machines could interact easily. It encapsulated all of its components as objects. This methodology has all the normal advantages of an object-oriented approach, such as data abstraction, encapsulation, inheritance and polymorphism. Legion was first released in November 1997. In August 1998, Applied Metacomputing, was established to exploit Legion commercially. In 2001, Applied Metacomputing was relaunched as Avaki Corporation.

In the third generation which is still developing today, the solutions involved increasing adoption of a service-oriented model and increasing attention to metadata. There is a strong sense of automation in third generation systems; for example, when humans can no longer deal with the scale and heterogeneity but delegate to processes to do so which leads to autonomy within the systems. These requirements can be seen very similar to the self-organizing and healing properties of biological systems. Therefore system has been termed autonomic after the autonomic nervous system.

The architecture that is accepted starting with second generation Grid systems consists of five layers [8](Figure 2.1).





**Figure 2.1:** Grid Protocol Architecture

The Grid *Fabric* layer provides the resources to which are opened for share by Grid protocols: A resource may be a physical entity such as a hard disk or a logical entity, such as a distributed file system, computer cluster, or distributed computer pool. Storage systems, computers, networks, code repositories, catalogs, sensors can be examples of *Fabric* layer resources. *For computational resources*, mechanisms are required for starting programs and for monitoring and controlling the execution of the resulting processes. At the same time some mechanisms are required for putting and getting files in *storage resources*. Some specialized form of storage resource requires mechanisms for managing versioned source and object code: for example, a control system such as CVS. *Network resources* must have management mechanisms that provide control over the resources allocated to network transfers (e.g., prioritization, reservation).

The *Connectivity* layer defines main communication and authentication protocols required for Grid-specific network transactions. Communication protocols enable the exchange of data between Fabric layer resources. Authentication protocols build on communication services to provide cryptographically secure mechanisms for verifying the identity of VOs and resources. Authentication solutions for VO environments should have the following characteristics: *Single sign on, Delegation, Integration with various local security solutions, User-based trust relationships*. Communication requirements include transport, routing, and naming. Communication (IP), service discovery (DNS), authentication, authorization, delegation are part of the *Connectivity* Layer.

The *Resource* layer comes over the *Connectivity* layer. *Resource* Layer defines communication and authentication protocols together with APIs and SDKs for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources. *Resource* layer protocols are concerned entirely with individual resources and therefore ignore issues of global state and atomic actions across distributed collections since these issues are the concern of the *Collective* layer. There are two primary classes of *Resource* layer protocols: *Information protocols* are used to obtain information about the structure and state of a resource. *Management protocols* are used to negotiate access to a shared resource, Specifying resource requirements including advanced reservation and Quality of Service and defining the operations to be performed, such as process creation, or data access are the examples of *management* protocols. Access to computation; access to data; access to information about system structure, state, performance are handled in this *Resource* layer.

While the *Resource* layer is focused on interactions with a single resource, *Collective* layer in the architecture contains protocols and services, APIs and SDKs that are not associated with any one specific resource but rather are global in nature and capture interactions across collections of resources. This is the reason why this layer is named as *Collective* layer. *Collective* layer owns services such as *Directory services, Co-allocation, scheduling, and brokering services, Monitoring and diagnostics services, Community accounting and payment services.*

The final layer in the Grid architecture is the *Application* layer which is started and operated by any VO that is in the Grid environment. Multidisciplinary Simulation, Ray Tracing, Crisis Management can be examples of Grid applications.

### **2.1.2.Integrating wireless devices into grid**

The integration of mobile wireless consumer devices into the Grid initially seems unlikely due to the inherent limitations of mobile devices, such as the low-bandwidth communication mentioned above, reduced CPU performance, small secondary storage and heightened battery consumption sensitivity. But the proxy-based clustered infrastructure solution provides mobile computing devices with favorable deployment, interoperability, scalability, adaptivity, and fault-tolerance characteristic

with the current abilities they have. And about the most important limitation, battery consumption, recent developments in lithium polymer replacements for lithium ion show promise in this field [9].

Wireless devices increasingly employ cameras, microphones, GPS receivers, and accelerometers. Another important class of devices is sensors, which can supply information on temperature, health monitoring and pollution levels. Being able to collect such data from anywhere, anytime shall give us a big area to develop interesting and useful applications to be used in Wireless Grid. In another view angle, by spreading workload across a large number of computers, the grid computing user can take advantage of enormous computational, storage, and bandwidth resources that would otherwise be prohibitively expensive to attain within traditional multiprocessor supercomputers [9].

The Integration of mobile devices into Grid architecture can be achieved by aggregation of three important research areas: Grid Computing, P2P networks and Web Services. And these technologies must be used in a proxy based clustered structure. Grid Computing is the basic idea that we shall somehow enable resource sharing and problem solving in dynamic, multi-institutional virtual organizations with wireless devices integrated in. Wired Grid is clearly a high-trust and relatively static environment. For example, Globus requires a machine wishing to participate in the Grid to create and pre-register an x509 certificate. P2P networks, such as Napster, Gnutella, and Kazaa, have characteristics in common with wireless grids. They must arrange coordinated sharing among heterogeneous devices, across unreliable network connections, with little or no prearrangement and little or no warning of site failure. Web Services are the key concept in remote access to resources. The Web Services technologies such as SOAP, lightweight XML message passing and the flexible WSDL, are very feasible to use in any ad-hoc sharing situations which are indeed apply to Wireless Grid architecture [10].

In the architecture there are two different categories of wireless devices. Laptops are in one category; PDAs, mobile phones are on the other. In [9] the words interlocutor for first category devices and minions for the second, are used. The same terminology will be used throughout this paper. One interlocutor and various numbers of minions will form a cluster. The interlocutor represents these devices to

the Grid on their behalf. Interlocutor contributes certain amount of computational, networking and storage resources. These resources are not just the ones it has, but also the available resources in the minions connected to interlocutor. With these abilities interlocutor can run Grid middleware such as Globus.

When a request comes to interlocutor from a resource consumer, it decomposes the request among the minions. It shall own a tool that will do the partitioning of the problem and data among these parallel computing units. The same tool shall process and aggregate data to come up with meaningful results when data is received from minions. Interlocutor shall also constantly monitor the available nodes under his responsibility. Then it can measure, maintain and report the required QoS even in the changing environments. To do this it must make necessary reservations on the resources [11]. Similar clustering mechanism is used in P2P network Kazaa by to facilitate scalable searching by Fast Track Infrastructure [10].

In order to keep communication and routing simple, the location information of minions must be cached by the proxy interlocutor. On the other hand in each minion, there shall be a component running that manages the overall resources such as the I/O bandwidth, batter power, CPU cycles and memory. This component will report the current availability of the corresponding resources once the minion has published the availability of its resources.

SOAP can be used in the communication between minions and interlocutor. The authentication in this client server mechanism can be established via GSI [11].

Once a Web Service is deployed in one minion, other applications and Web Services can discover and invoke that service. If Information Resources of a wireless device and device parameters can be made as “Web Service” stored in a Distributed UDDI registry with descriptions about usage in XML. Then kind of “yellow pages” are published as a “Web Service” in a distributed UDDI for Midlet groups.

The interlocutor advertises the wireless device’s available information resources as grid services via OGSA’s Indexing Service mechanism. It also should be able to consider the order in which jobs are submitted and what nodes are available; it should also maintain the required QoS during any environmental changes. It makes

and enforces the necessary resource reservations when it receives a request from a grid node. The proxy terminates or cancels these reservations once the requested tasks are completed or if the node can't offer its resources for sharing. The communication mechanism shall give support for these actions.

The interlocutor and minions can communicate via SOAP and authenticate each other via the Generic Security Service (GSS) API. The interlocutor analyses code and checks for resource allocation through the Monitoring and Discovery Service (MDS).

After the interlocutor determines resource availability, the middleware layer component in the server sends the job request to remote locations. Grid system returns a report about the execution's success back to the interlocutor, which then directs these results to the mobile node via the router. The router redirects the packets according to the mobile node's Care Of Address (COA), thereby giving the user the ability to roam. Of course caching of mobile nodes locations can give a real help to the server. Prefetching and caching decreases the resource connection's instability in the wireless grid environment. So it improves the application execution's performance [11].

As mentioned before SOAP is the communication platform between interlocutor and minions. SOAP is based on XML messages. But for mobile communication kXML and kSOAP are introduced [13]. kXML is the XML pull parser and writer suitable for Java 2 Micro Edition (CLDC/MIDP/CDC). It has a small footprint size that it is especially suited for Applets or Java applications running on mobile devices like PDAs or MIDP enabled cell phones. On the other hand kSOAP is an SOAP API suitable for the Java 2 Microedition based on kXML. Because of its small footprint it may be suitable to build SOAP-enabled Java Applets as well. Moreover Remote Procedures Calls can be easily used with MIDP devices with kSOAP parser. There are also some other approaches for SOAP to be used in wireless infrastructure such as Wireless SOAP. WSOAP offers a new encoding and synchronization technique between wireless device and the gateway to deliver significant bandwidth and packet loss reduction [14].

If we get into discovery mechanism details, web services are deployed by the mobile devices and then these services can be discovered and invoked by other applications and services. These services will form the “yellow pages” directories for roaming mobile devices. Middleware components and clients can turn a wireless device’s information resources such as hardware and platform types or performance parameters into Web services and store them in a UDDI registry with XML descriptions about its usage and the services it offers. Distributed registry is accessed and updated for various types of devices and resources.

To support service discovery, the distributed registry can be extended to include QoS-related parameters with which the bandwidth broker can translate services based on service guarantees. Since it is a wireless grid system there shall be some extra mechanism and communication in order to keep the system survivable. This mechanism can also make the system secure and scalable. This can be established by extending the distributed registry to include some Quality Of Service related parameters. A bandwidth broker implemented in a wireless grid proxy can manage QoS parameters within a given wireless domain based on the authorized resources available in that domain.

The bandwidth broker in interlocutor can also manage interdomain bandwidth reservations to coordinate grid nodes in neighboring domains and communicate with local grid node monitoring tool to determine network state [11].

To see how the QoS can be achieved we can give a typical application scenario. A minion contacts with its proxy about its request. Interlocutor then queries its authorized registries that have the specified service with specified QoS capabilities within any domain. The registries responds by sending a list of matching services to the proxy to determine which one is best suited for the client’s needs. Then proxy contacts the specified service domains’ bandwidth broker and middleware for resource availability. After resource availability is understood, job scheduler gets invoked, bandwidth broker facilitates and conducts resource allocation functions. Then selected service can be initiated for execution. During the execution interlocutor constantly monitors QoS status to signal any change in the parameters. Figure 2.2 shows QoS specification XML example that can be used to send by the minions to the interlocutor about its QoS status.

```

<Service>
  <CPU-QoS> 10 CPU </CPU-QoS>
  <Memory-QoS> 64 MB </Memory-QoS>
  <Network-QoS>
    <Wireless-Device> Siemens-SXG75 </Wireless-Device>
    <Bandwidth-Rec> 115 Kbps </Bandwidth-Rec>
    <Packet-Loss> Less Than 2% </Packet-Loss>
    <Delay> 10ms </Delay>
  </Network-QoS>
</Service>

```

**Figure 2.2** : QoS specification XML message

But how will the wireless communication survive in network failures? Alastair Hampshire from University of Nottingham proposed extending Open Grid System Architecture to Intermittently Available Wireless Networks. [15]. His framework uses intermediary SOAP Routers to queue messages during periods of network disconnection as well as providing finer grained retransmission. Like Snoopy approach in Wireless TCP, there are agents called SOAP routers that queue the SOAP messages and in case of disconnection, the retransmission takes place not end to end but from the SOAP router to the receiver.

### 2.1.3. Wireless grid applications

There are some applications developed using by integrating wireless mobile devices into Grid, even these applications are just far beyond the potential usage.

DARC (Distributed Ad-Hoc Resource Coordination) is a project that is carried out in Syracuse University. It lets devices with no prior knowledge of each other collectively record and mix an audio signal such as a concert, speech, lecture, or emergency event [3].

There is a research carried out in Helsinki Institute of Physics called Gridblocks [4] that is focused on integrating Wireless Java devices into Grid. An healthcare application is developed that can detect the movement of lungs and heartbeats with 90 percent certainty within 50 cm distance of human body to produce valuable information of pulse and breathing rate [3].

Mobile nodes in war zones can broadcast high data information in a secured, authenticated environment through use of the Grid network in the same army battalion. Also, the proxy through its use of caching can effectively handle identical

or repeated requests in limited time intervals specified. The proxy can also communicate to other proxies defined in other battalion camps to map effective routes of communication with the Grid network for job submissions [11].

The AKOGRIMO project [5] is a European funded project aiming to architect and prototype Next Generation Grid based on Open Grid Services Architecture (OGSA) which exploits and closely co-operates with evolving Mobile Internet infrastructures based on IPv6. The concept of the project is to evaluate the derived Mobile Grid through testbeds that are chosen based on existing evolving technologies. The applications that are worked on or planned to be worked are e-Health, e-Learning and Crisis management.

The GridLab project [6] is one of the biggest European researches in the development of application tools and middleware for Grid environments. They are working on a workpackage to develop theoretical scenarios and their implementation for using mobile devices to enhance the working environment and efficiency of application users. This workpackage is trying to make Portal, web interface, remote steering and visualization to be supported on a wide range of mobile devices such as PDAs, laptops and mobile phones.

## **2.2. Wireless Devices**

Wireless devices and communication alternatives are developing everyday but their limitations will continue to be there.

### **2.2.1. Wireless devices and their limitations**

Mobile devices penetrated in everyone's life without national, geographic or economic limits. We can divide these mobile devices in two main categories for our approach: laptops on one category, PDAs and mobile phones on the other. The participations and benefits of these devices in Grid can differ according to these two categories.

Laptops are taking places of PCs. These devices have fast processors and sufficient secondary storages. Even seti@home project may not have been considered as Ideal



Grid application, its success in making PCs participate in such an organization can be a good example for usage of laptops in Grid. In a proxy based architecture that will be given, laptops can serve as proxy servers. Today an average laptop comes with two processors each having 1.80 GHz processing power, 1024MB memory, 80GB hard disk and integrated 802.11 b/g wireless connection card. Using an external GSM/GPRS modem card or modem function of a mobile phone, they are not limited to 802.11 Wi-fi areas but rather can get connected anywhere with GSM networks.

Although PDAs and mobile phones have different hardware and software, in our approach for Grid, they will be considered in same category. These devices will be connected to proxy servers in order to participate in Grid. The reason is these devices can not run a middleware on them. For instance, the grid portal Globus requires substantial memory and storage requirements, which eliminates possibilities for wireless devices to run a Globus API [11].

As Moore's Law indicates, increasing transistor density results in increased CPU performance of PDAs [9]. Average PDAs own 416 MHz processor and 128MB RAM memory. Windows Mobile operating system is widely used on them. They are equipped with wide touch screens, cameras, GSM modules, 802.11b/g cards and Java.

Mobile phones are also developing very fast that special operating systems such as Symbian and Windows Mobile are being developed that can run on devices from different manufacturers. The common development platform in these devices is the Java 2 Micro Edition (J2ME) which enables developers to implement applications independent of the manufacturers and operating systems.

Manufacturers are trying to add new features to PDAs and mobile phones to create new demands for new products within small time intervals. Together with some other functions and hardware, GPS receiver is the new popular addition to these devices. Siemens SXG75, BenQ-Siemens P51, Nokia N95, Nokia E90, Nokia 6110, Motorola A1010, HP HW6915, NEC E616, Blackberry 8800, i-mate JAQ4 are some examples with GPS receiver that can be find on shelves of the phone markets. The number will increase more, since even rather a smaller company like BenQ-Siemens has three models with GPS receiver on their 2007 roadmap [16].

The integration of mobile wireless consumer devices into the Grid initially seems unlikely due to the inherent limitations of mobile devices, such as the low-bandwidth communication mentioned above, reduced CPU performance, small secondary storage and heightened battery consumption sensitivity. But the proxy-based

clustered infrastructure solution provides mobile computing devices with favorable deployment, interoperability, scalability, adaptivity, and fault-tolerance characteristic with the current abilities they have. And about the most important limitation, battery consumption, recent developments in lithium polymer replacements for lithium ion show promise in this field [9].

Wireless devices increasingly employ cameras, microphones, GPS receivers, and accelerometers. Another important class of devices is sensors, which can supply information on temperature, health, or pollution levels. Being able to collect such data from anywhere, anytime shall give us a big area to develop interesting and useful applications to be used in Wireless Grid. In another view angle, by spreading workload across a large number of computers, the grid computing user can take advantage of enormous computational, storage, and bandwidth resources that would otherwise be prohibitively expensive to attain within traditional multiprocessor supercomputers [9].

### 2.2.2. Wireless communication alternatives

While most of laptops and PDAs offer wireless cards embedded that can connect to 802.11b/g networks, others shelter a slot for an external card. We assume that minimum 11Mbps offered by wireless networks would be enough for proxy servers to be able to run middleware and participate in Grid fully. If data rate is not very essential then GSM/GPRS modem cards or modem functionality of mobile phones can be used after they are connected to the laptops to have connection to the internet over GSM networks. This would remove the limitations on the laptops mobility to stay in the 802.11b/g network areas. GSM functioning PDAs and mobile phones can get connected to Internet with different speeds depending on the GSM operator infrastructure but minimum with a 160Kbps data rate. Bandwidth and ranges for different wireless technologies are given in Table 2.1.

**Table 2.1:** Bandwidths and ranges for wireless technologies

	<b>802.11b</b>	<b>802.11g</b>	<b>Bluetooth</b>	<b>GPRS</b>	<b>EDGE</b>	<b>UMTS</b>	<b>HSDPA</b>
<b>Bandwidth</b>	11Mbps	54Mbps	1Mbps	160.0Kbps	236.8Kbps	2Mbps	3.6Mbps
<b>Range</b>	100m	100m	10m	Cellular	cellular	cellular	cellular

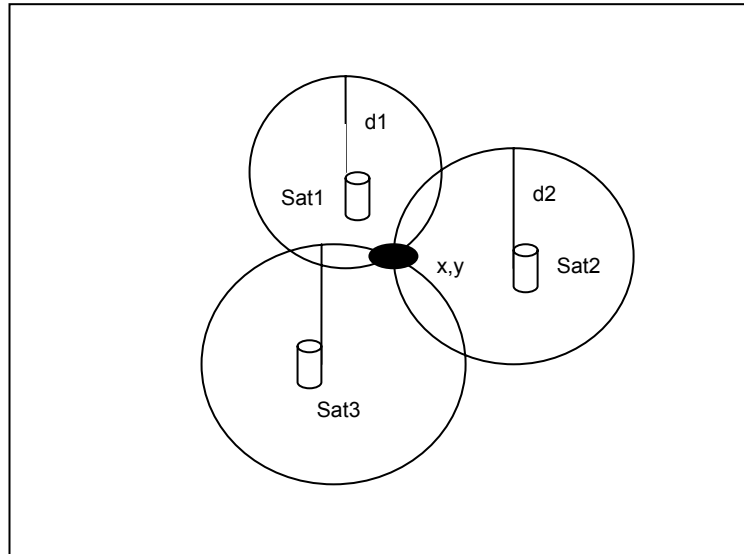
### 2.3. Global Positioning System

The Global Positioning System (GPS) is a space-based radio positioning system designed to provide suitably equipped users with highly accurate positioning and time data. GPS provides high accuracy, worldwide coverage, all-weather operation and usefulness at high velocities.

18 satellites in well-chosen orbits would provide worldwide coverage. There are three more included as spares, orbiting on-call, ready to replace malfunctioning and obsolete units. More satellites orbiting increase reliability, readings and accuracy. All 24 satellites are placed in six very high orbital planes, from lowest altitude 20.200 km to highest altitude 20.300. Satellites have an orbital period of 11 hours and 58 minutes. GPS satellites orbits are tilted at an angle of 55 degrees relative to equator therefore at least four satellites are in view above the horizon at any time. All GPS satellites have two frequencies: L1 (1575.42 MHz) and L2 (1227.60 MHz)

The calculation of position is done by measuring the distances to at least three satellites from that position. This process is called *triangulation*. If you draw circles with radiuses equal to the distances from satellites, these circles would meet at one point. This point is the position that we measured the distances from [17]. This is illustrated in Figure 2.3. Enough information may be obtained from 3 satellites to determine latitude and longitude but if altitude is also requested then 4 satellites are needed.

A GPS receiver can measure the distance from satellites by one of the basic formulas in Physics:  $Distance = Speed \times Time$ . The radio signals travel in the speed of light. So the time it takes from the satellite to the receiver shall be known. Time that the signal is transmitted from the satellite is encoded on the signal, using the time according to an atomic clock on the satellite. Time of signal reception is recorded by receiver using an atomic clock. So the time difference is multiplied by the speed of light yield the distance traveled by the signal.



**Figure 2.3: GPS Triangulation**

The accuracy of a specific model of receiver is determined by the receiver's ability to receive specific radio signals being sent from the satellites, how the receiver uses the signals it is receiving, the software program used to calculate the location of the receiver, the amount of internal noise of a particular receiver.

The Pseudo Random Code (PRC) is a fundamental part of GPS. Physically it's just a very complicated digital code, or in other words, a complicated sequence of *on* and *off* pulses. The signal is so complicated that it almost looks like random electrical noise. Therefore the name is Pseudo-Random. There are several good reasons for that complexity: First, the complex pattern helps make sure that the receiver doesn't accidentally sync up to some other signal. The patterns are so complex that it's highly unlikely that a stray signal will have exactly the same shape. Since each satellite has its own unique Pseudo-Random Code this complexity also guarantees that the receiver won't accidentally pick up another satellite's signal. So all the satellites can use the same frequency without jamming into each other. And it makes it more difficult for a hostile force to jam the system. But there's another reason for the complexity of the Pseudo Random Code, a reason that's crucial to making GPS economical. The codes make it possible to use information theory to amplify the GPS signal. And that's why GPS receivers don't need big satellite dishes to receive the GPS signals [17].

There are 3 distinct segments of GPS: Space, control and users. Space segment is comprised of 21 working satellites and 3 spares. But even the average lifetime of a satellite calculated as 7.5-10 years, some of old ones continue to work fine. Therefore as November 2005 there are 30 GPS satellites working. These satellites are sent by U.S. military but open for public use. They are arranged in six nearly circular orbits tilted at an angle of 55 degrees relative to the axis of the earth. The number and orbits are chosen to have at least 4 satellites in any open reception point on earth [18]. The satellites are equipped with atomic clocks, computers, sensors, batteries, solar array panels, maneuvering jets and antenna. Control segment is composed of Master Control Station (MCS) based on Colorado, US and monitoring stations around the world. MCS predicts satellite orbits, keeps satellites on these orbits, and activates spare satellites when needed, calibrates on board clocks, checks the batteries and solar panels, monitors all satellite transmissions and updates the navigation message. Monitoring stations continually tracks all satellites in view and forward their data to MCS. The user segment of GPS includes anyone who captures the transmissions of the GPS satellites with necessary receiver equipment [19].

There are three types of GPS signals: P-code, C/A-code and Y code. P-code is the precise code that is used for military purposes. P-code is transmitted at a high bandwidth, hard to acquire and less susceptible than civilian code to being spoofed. C/A code is the civilians code which is relatively short sequence of numbers sent on a relatively narrow bandwidth. Y-code is available only to users with special deciphering equipment – military receivers with a special chip installed.

### **2.3.1. Applications**

There can be three main application areas using GPS: Industry, military and science. Navigation, tracking, agriculture, mapping and GIS data collection, public safety, surveying and telecommunication applications can be produced in industry. Military applications can include intelligence, target location, navigation, weapon aiming and guidance. Lastly scientific researches on areas such as archaeology, atmospheric science, environmental, geodesy, geology, geophysics, oceanography and wildlife can use applications that use GPS.

### **2.3.2. Future of global positioning system**

There are some changes planned in the GPS in the near future. Between 2006 and 2010 Block IIF/GPS IIF type new satellites will be placed on the orbits. The L5 signal on 1176.45 MHz will be used on these satellites. Block III satellites are still in design process and planned to be placed on orbits after 2012. The biggest differences of them are that they are powered with well anti-spoofing and anti-jamming mechanisms which are very important for military purposes.

There are some other satellite systems developed for positioning. GLONASS was founded by former Soviet Union in 1970s and is a working system right now controlled by Russia. If both GPS and GLONASS can be used together, more accurate and quick responses are possible for positioning defining. Therefore even now GPS + GLONASS functioning receivers are started to be produced [18].

Galileo is another positioning satellite system alternative came out against GPS originated at Europe. The design of the Galileo started in 2001 and it is planned to place as a total of 26-28 satellites in orbits by the end of 2007. It is expected to Galileo start servicing in 2008 [18]. Hopefully Turkey can be a part of the Galileo satellite system.

With all these changes and alternatives we will have more accurate and fast positioning systems which will increase the number of applications that will be involved in this area.

### **2.4. Motivation**

In observing the nature, giving direction to nature, reaching the far space, foreseeing the future, modelling everything important for human life, managing crisis and solving every problem, humankind is more willing than anytime before. The everyday developing technology is allowing him to reach some of his goals. But for the success in the above listed topics and unlisted others, we think that Grid technology which is predicting sharing of resources will play an important role.

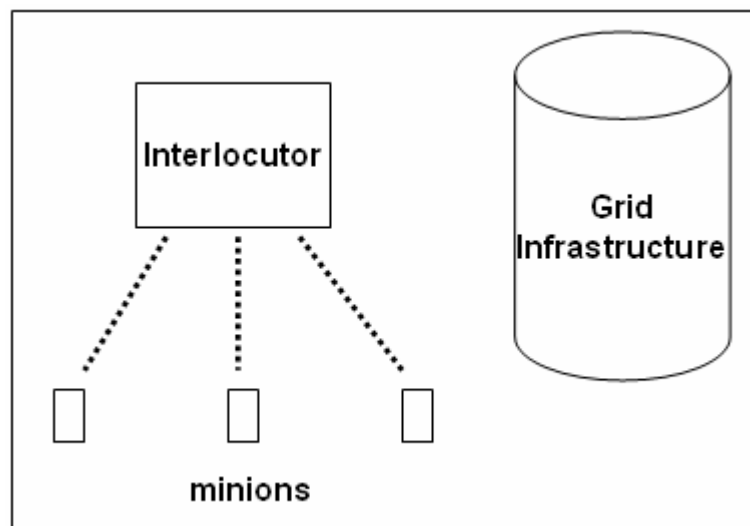
In stead of limiting Grid with just powerful parallel processors, huge storage units and strong visualization centers, adding it the human itself together with personal hardware and technologies, can give lots of other meanings and different aspects to Grid. Together with the PCs that people have in their houses and offices, they own

hardware that they always keep near and almost permanently connected: mobile phones and GSM networks. At the end 2005 there was around 1 mobile phone for every three inhabitants on the planet, with the total number reaching 2.14 billion.[20] It is possible to imagine the number reached today. The number of mobile phones in the world is far ahead of internet adoption. There will be great profits if we can add mobile phones and other mobile devices with their everyday developing nature, as resources to the Grid. Perhaps it is not very far to see the days that whole planet people will vote for United Nations Secretary election in a worldwide democracy with just a few keypad clicks developed with a Grid infrastructure.

Starting with these ideas after researches on the corresponding areas, it was decided to work on a project to show how wireless mobile devices can be integrated into Grid technologies and the IMOGA architecture is suggested. A sample application that can use the data collected from ubiquitous mobile phones with GPS capability is implemented. We believe that this project can be expanded to environment and health areas with sensors attached to the mobile phones.

### 3. THE IMOGA AND CIM-TR

In this project we suggest that a noble architecture called IMOGA (Integrating Mobile Devices into Grid Applications) to address the issues of integrating mobile devices into a Grid system and sharing and producing information in a cooperative manner. There are two different categories of wireless devices. Laptops are in one category; Personal Digital Assistants (PDA), mobile phones are on the other. For the first category devices *interlocutor* and second category devices *minion will be used* as it is given in section 2.1.2. One interlocutor and various numbers of minions form a cluster. The interlocutor acts as a proxy and represents minions to the Grid so that the system can be assembled by utilizing minions possessing even limited communication and computation power. The interlocutor has to be powered with a certain amount of computational, networking and storage resources given that these resources are not just the ones it has, but also the available resources in the minions connected to interlocutor. With these abilities interlocutor can run Grid middleware or have a mechanism to connect with a middleware (Figure 3.1).



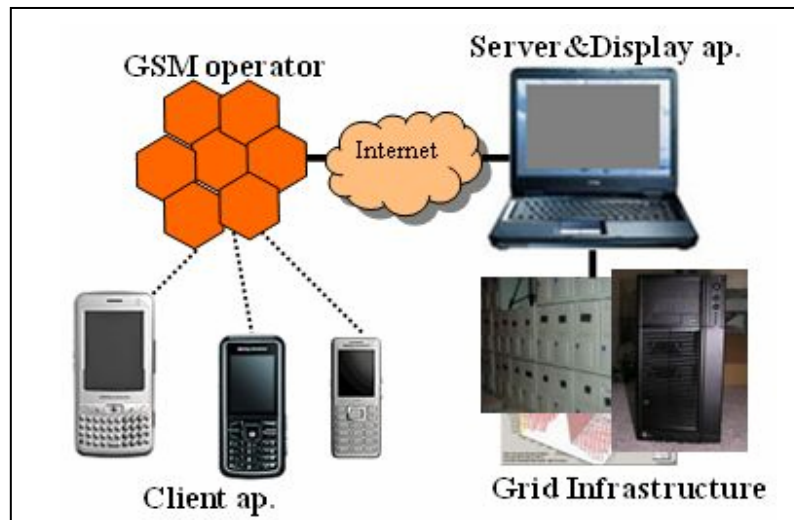
**Figure 3.1:** IMOGA architecture



Although PDAs and mobile phones come with different hardware and softwares, they will be considered in the same category (i.e. minions) since they share the same communication protocols. These devices will be connected to interlocutors in order to participate in Grid since they can not run a middleware on them. Note that the well known Grid portal called Globus requires substantial amount of memory and storage, which eliminates possibility of minions to run a Globus API [11].

Mobile phones are the basic minions. They are developing very fast that special operating systems such as Symbian and Windows Mobile are being developed. Such operating systems can run on devices from different manufacturers. A common feature of these devices is the support for Java 2 Micro Edition (J2ME) which enables developers to have a common platform for their applications. Besides cameras and microphones mobile phones increasingly employ new devices such as GPS receivers and accelerometers. Moreover there is a possibility of attaching sensors to the mobile phones which can supply data on temperature, personal health, environmental pollution etc. These sensors can be easily connected via Inter Integrated Circuit I<sup>2</sup>C protocol [21] or Bluetooth®. Being able to collect such data from anywhere, anytime, shall give us a big area to develop interesting and useful applications to be used in Wireless Grid. The traffic application produced for IMOGA architecture is based on mobile phones with GPS capability on board.

In this paper the IMOGA architecture is tested against an application that collects data ubiquitously and produces information for the very same data collection nodes. In this application (Cooperative Information Management for vehicle TRaffic flow CIM-TR) minions transfer their location and speed information to the interlocutor and interlocutor processes the received data and provides traffic information to data providers as well as the public. CIM-TR architecture is given in Figure 3.2.



**Figure 3.2** CIM-TR architecture

CIM-TR is composed of three main parts. *Client* application runs on the minions so that they can produce and transfer their location and speed information to the interlocutor. The produced data are sent to the interlocutor in SOAP XML messages. Parameters to be passed in the XML are chosen to be strings because results of the test runs indicated that transferring data in the forms of strings lead to the minimum cost and maximum flexibility. The client application has also a basic interface which shows longitude, latitude, speed and some other parameters that are used in communication mechanism.

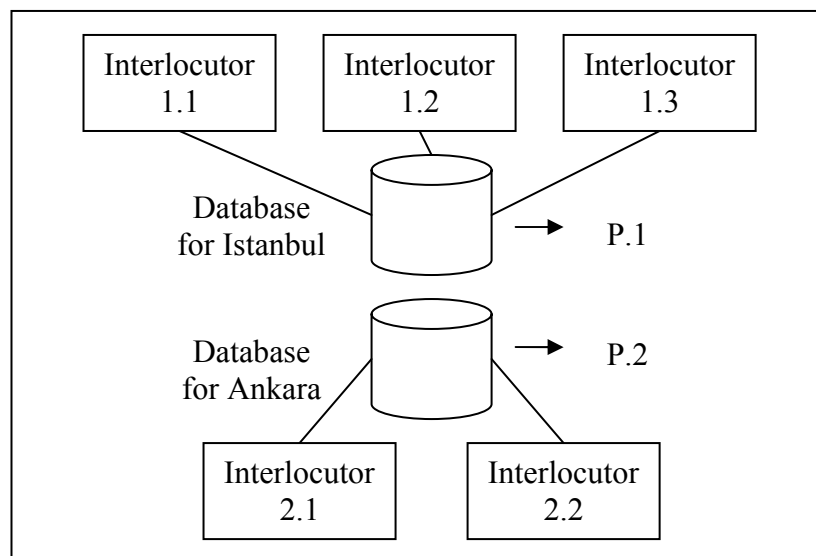
The second main part *Server* application works on interlocutor and is planned to communicate with a Grid middleware. It has an access to a MySQL database created for Server and Display applications. A main road database is created according to GDF specifications [22]. with some variations. GDF is a European standard that is drawn up by Comité Européen de Normalisation in co-operation with digital map providers, automotive and electronic equipment manufacturers, to describe and transfer road networks and road related data. For each location data received from the minions, it is checked to be on the roads that are in the database. If the location comes out to be on a road, new speed information is recorded with a timestamp.

Minions and interlocutors communicate each other via three-tier Web Service Client mechanism produced by Java. The three "tiers" are client, web application containing

servlet, and server. The client communicates with the generated middleware servlet and the servlet communicates with server.

The last part of the CIM-TR is *Traffic Calculation and Display* application which is a Java applet that connects to the database to retrieve user locations and speed information and display it to the public. It has two main functions. One is to show the location and speed of minions on a map which can be helpful if we would like to offer free tracking system to the contributors. Second is to calculate the average speeds on the roads and display it. Application uses statistical data when there is no such data available. The map covers the city of Istanbul and the coordinates of the main roads of Istanbul such as E5 and TEM are entered in the database for now. But it can be expanded to other cities and roads since the database and applications are implemented to be flexible.

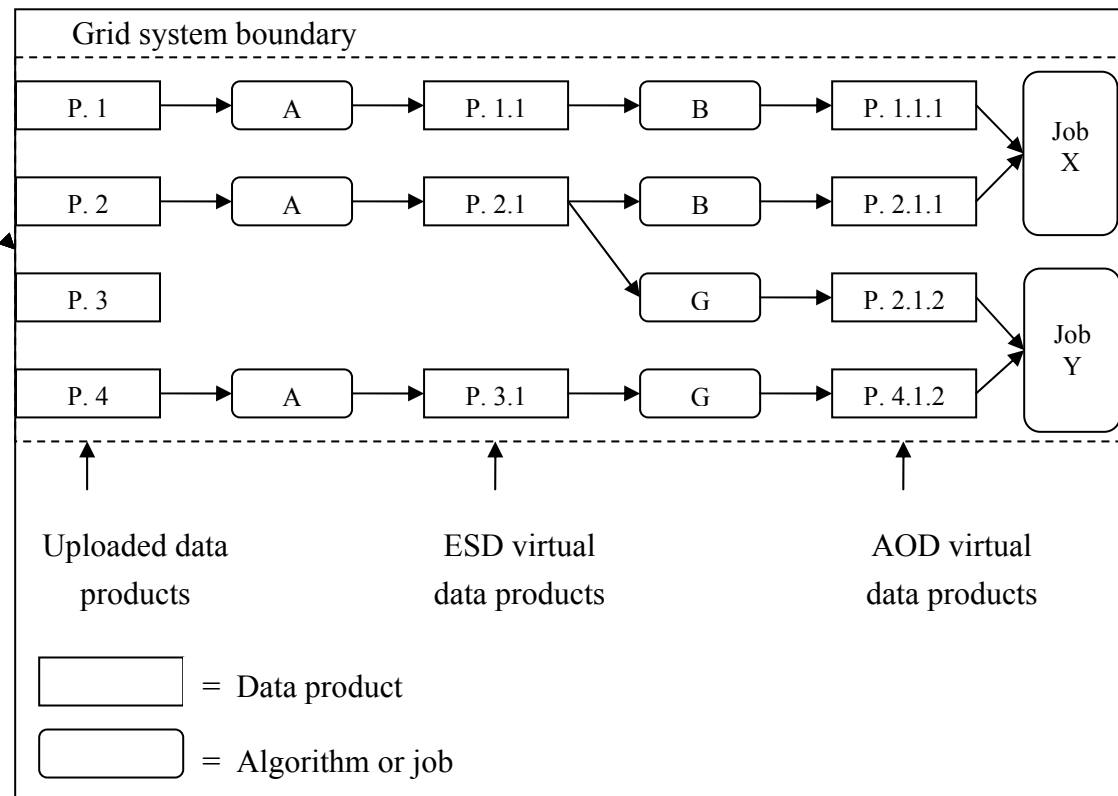
How can this data produced by the IMOGA be used in a Grid system? Now let's look at a sample Grid system which can use the data produced by the various interlocutors. There can be different interlocutors at different locations. These interlocutors can add new data to the city databases such as İstanbul and Ankara. This each city database can be the uploaded data products for the Grid system Figure 3.3.



**Figure 3.3** Uploaded Products by Interlocutors

The Grid system given in Figure 3.4, is defined as the HEPGRID2001 data model [23] which can also be used for IMOGA. Any piece of data in the Grid system is called a “data product” in the model. A data product is the smallest piece of data that the grid needs to handle.

There are two types of data products: uploaded and virtual. The uploaded data product contains the speed information produced by the server application for single cities. A virtual data product is the output of some data processing algorithm that has been registered with the Grid. Having the algorithm registered with the Grid, virtual product can be computed on demand. The algorithm will use the value of another virtual or uploaded data product as its input. Each product has a unique identifier and with these identifiers of virtual products, grid can determine which algorithms and inputs are needed to derive the product value. The virtual data products obtained directly from the uploaded data products are generally called Event Summary Data(ESD). And those obtained from ESD products are called Analysis Object Data (AOD) products.



**Figure 3.4** Sample Grid System

As an example in the Grid System given in Figure 3.4, algorithm A may calculate approximate amount of pollution caused by the exhaust of the cars on the city highways. Using the data product P.1.1, algorithm B can calculate the effect of city highways on global warming by calculating the amount of greenhouse gases. As a result Job X can use the product P.1.1.1 to report the results to the Environment Ministry of the countries.

## 4. IMPLEMENTATION OF CIM-TR

The implementation of CIM-TR is composed three main parts. *Client* application which runs on mobile phones, *Server* application which is the core of CIM-TR and *Traffic Calculation* and *Display application* are the main parts which are explained in details below. The three tier web service client communication mechanism between *Client* and *Server* is also given in this chapter.

### 4.1. Client Application

*Client* application is a Java MIDlet which runs on mobile devices with J2ME MIDP support and GPS receiver hardware. Application acquires date, latitude, longitude and speed values from GPS receiver with JSR 179 Java location API. Together with these data, IMEI unique identification number is also passed to the server. The acquired data are sent to the server in SOAP XML messages. SOAP XML messaging protocol provides various data formats to send this data.

#### 4.1.1. Decision making on XML parameters

Since minimization of data transfer cost is essential due to GPRS charging policy of GSM operators, we have done a number of experimental runs to choose the data format that will cost less.

In the first experiment, *Client* was designed to send two double values as longitude and latitude, two strings as IMEI and datetime. There were three different versions established:

- 1) Reads and sends the data in every 2 seconds

*Message e.g.*

IMEI:357250000120006 loc\_date:Thu Nov 23 09:10:49  
GMT+02:00 2006

lo:29.10075008869171 la:40.971402525901794

**2) Reads and sends the data in every 10 seconds**

*Message e.g.*

IMEI:357250000121848 loc\_date:Thu Nov 23 09:09:49  
GMT+02:00 2006

lo:29.101635217666626 la:40.96993267536163

**3) Reads data in every 2 seconds but sends data in every 10 seconds**

*Message e.g.*

IMEI:004400017455120 loc\_date:Thu Nov 23 09:11:05  
GMT+02:00 2006

lo1:29.100374579429626 la1:40.972250103950500  
lo2:29.100240468978882 la2:40.972518324851990  
lo3:29.100068807601930 la3:40.972861647605896  
lo4:29.099982976913452 la4:40.973129868507385  
lo5:29.099859595298767 la5:40.973360538482666

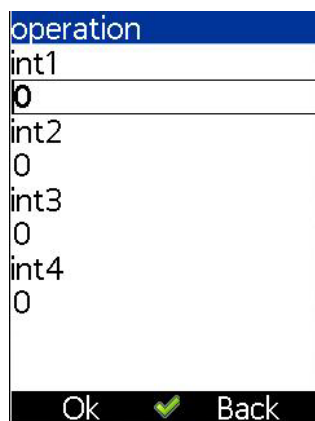
These clients ran on three different mobiles for about 22 minutes traveling about 10 km. As it can be seen from the results on Table 4.1, even if 1<sup>st</sup> and 3<sup>rd</sup> versions send the same amount of location data except that 1<sup>st</sup> sends timestamp for each location whereas 3<sup>rd</sup> sends 1 timestamp for all 5 locations , there is a big difference in the cost. So if we could send the time interval between location readings then timestamp for each location reading in the 3<sup>rd</sup> version gives the minimum cost with maximum data.

**Table 4.1:**Results of client versions in experiment 1

	<b>Number of messages sent</b>	<b>Data sent (KB)</b>	<b>Data received (KB)</b>
<b>1</b>	578	317	181
<b>2</b>	82	47	27
<b>3</b>	123	95	40

XML gives us the flexibility to send different type of parameters. To see the difference in byte costs of parameters in a message there were 4 different versions with different message types are implemented:

- 1) One string parameter: 50 characters are enough to send one location information.
- 2) One double parameter: The maximum value of double is  $2^{1023}$  in Java. One double value seems enough to send one location information.
- 3) Four integer parameters: Figure 4.1. The maximum value of integer is  $2^{31}$  in Java. We need approximately four integer values to send one location information.



**Figure 4.1:** The user interface of 3<sup>rd</sup> client version in 2<sup>nd</sup> experiment

- 4) Two string and two double parameters: This was the original message type that has been used before in experiment 1. It used to see the difference with other messages.



5) **Table 4.2:** Results of client versions in experiment 2

	10 Mes. Sent (First run)(KB)	10 Mes. Received (First run)(KB)	10 Mes. Sent (Second run) (KB)	10 Mes. Received (Second run) (KB)
1	4	3	4	3
2	4	3	6	4
3	4	3	4	3
4	4	3	5	3

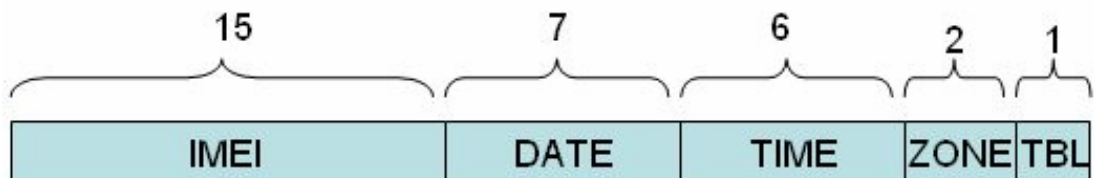
There were two test runs for each client. One test run was composed of sending and receiving 10 messages. The results given in Table 4.2 showed that even there is not much difference between passing different parameters, double is a little bit more costly.

As a result of the two experiments explained above it was decided to use the mechanism given below that packs data as two strings. Packing and parsing of strings will be much easier compared to integers, knowing that original data is composed of date, string, and doubles.

#### 4.1.2. Message structure

The message is composed of two parts: *header* and *location\_data*. The header has a fixed length whereas *location\_data* is a variable.

The *header* is composed of the following fields as shown in Figure 4.2.



**Figure 4.2:** The fields in the header

*IMEI*: 15 characters. It is the identification number for all cellular phones which is unique for each device. It will be used as identification for messages also in our communication.

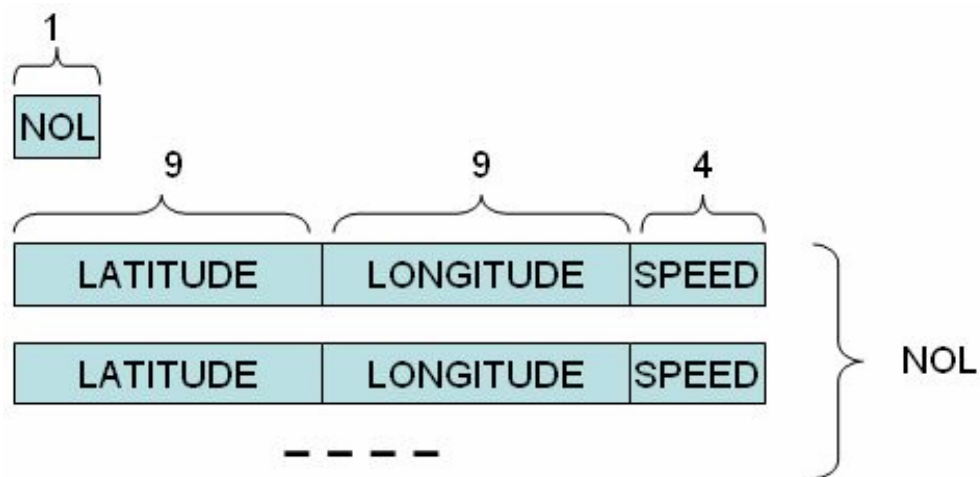
*DATE*: 7 characters. Date is in DDMonYY format

*TIME*: 6 characters. Time is in HHMMSS format

*ZONE*: 2 characters. Timezone is important for international speed calculations. It is also required in countries using different time zones such as USA.

*TBL(Time Between Locations)*: 1 character. Since there can be more than one location information in one message, it is required to show the time interval between these location readings. On server side it will be used to calculate the exact time for each location reading. It shows time interval in seconds.

*Location\_data* is composed of the following fields(Figure 4.3).



**Figure 4.3:** The fields in the *location\_data*

*NOL (Number Of Locations)*: 1 character. It shows the number of location readings that will be sent in a single *location\_data*. It gives us the flexibility on number of readings to be sent in a single message.

*LATITUDE*: 9 characters. It is the latitude of the mobile device. By using the standard precision representation 9 characters is shown to be sufficient to have a precise location information in the form of DD°MM'SS.SS".

*LONGITUDE*: 9 characters. It is the longitude value of the mobile device. Similar to the latitude parameter 9 characters would be sufficient for representation.

*SPEED*: 4 characters. Most of the GPS receivers in the mobile devices automatically produces a speed value. For those of which does not produce will require a simple mathematical calculation to produce this value.

*TBL* and *NOL* parameters provides us flexibility over cost and efficiency. Values of these parameters are fixed in our experiment but they can be programmed to be adaptive according to the speed of the mobile device.

#### 4.1.3. User interface

The client application has also a basic interface. This interface shows longitude and latitude in degree, minute, second format (Figure 4.4).



**Figure 4.4:** User interface of client application

As the GPS receiver in mobile device returns latitude and longitude values in double format, the application converts these values to degree, minute and second format. The client application also indicates speed and the state of GPS receiver that can be either *available*, *out of service* or *temporarily unavailable*. Since update interval of locations and the number of location information in one message are also variables for cost estimation they are displayed in via the interface. Every time the application sends a message, "*MESSAGE SENT*" indication is displayed on the client interface.

## 4.2. Server Application

Server application runs on a Sun Java System Application Server 9 which is open to the internet. The server application has to be connected to the internet since the client application messages are delivered through internet via GSM operator. This application works on interlocutor and is designed to communicate with a Grid middleware. Database operations are implemented locally through MySQL.

There is a Web Service on this server opened to the internet. This Web Service is called by the mobile phones via the communication mechanism that will be explained later. Server application receives the parameters sent in the messages by the mobile phones. It checks the length of the *header* and *location\_data* parameters and if the parameters are valid, it parses the different fields packed in the parameters. The exact time for each location data is calculated with the help of TBL. The location data for the IMEI number is recorded in the *Location* table which has the fields given in Table 4.3. This lets us to offer vehicle/user tracking to the contributors of the IMOGA system. Then the next step is to find the street that the location coordinate data is coming from.

The main purpose of the server application is to locate the street that the vehicle is moving along and record the received speed information along with a timestamp for that street. To locate the streets that vehicles are moving along, the coordinates of the streets are needed. Here there were two possibilities, one was to use a commercial digital map and second was to implement a basic map for at least main highways of İstanbul.

**Table 4.3:** Location table in the database

Location
Location_ID
IMEI
location_date
location_time
location_timezone
latitude
longitude
speed

Basarsoft, Navturk companies from Turkey, AND Company from Netherlands and Istanbul Municipality had digital maps of Istanbul. But since to purchase these maps, big budget was required, it was decided to implement a basic map for main highways of Istanbul.

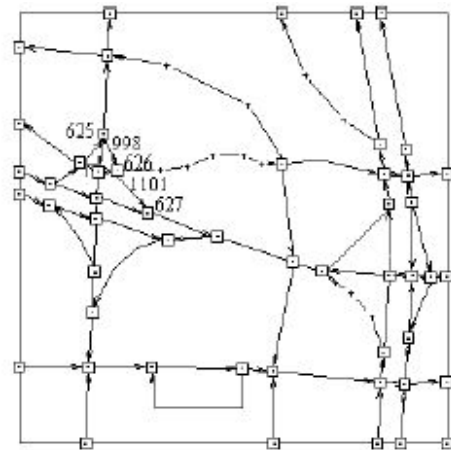
Following a survey on digital map formats and standards such as GDF, Federal Geographic Data Committee (FGDC)[24], ISO 19110 [25]; it was decided to implement a mapping system based on GDF.

#### 4.2.1. GDF

GDF [24] is a European standard that is used to describe and transfer road networks and road related data. It was drawn up by Comité Européen de Normalisation in cooperation with digital map providers, automotive and electronic equipment manufacturers. GDF has a three-level structure:

*Level 0: Topology.* This is a common Geographical Information Systems (GIS) topology description that everything has been described by nodes, edges and faces (Figure 4.5).

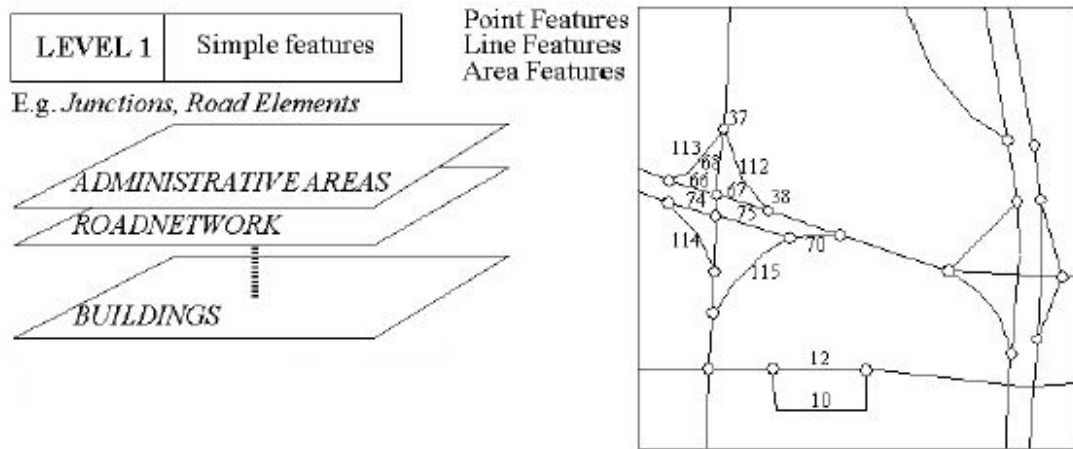
<b>LEVEL 0</b>	Topology
Nodes, Edges, Faces	



. **Figure 4.5:** GDF, Level 0, Topology

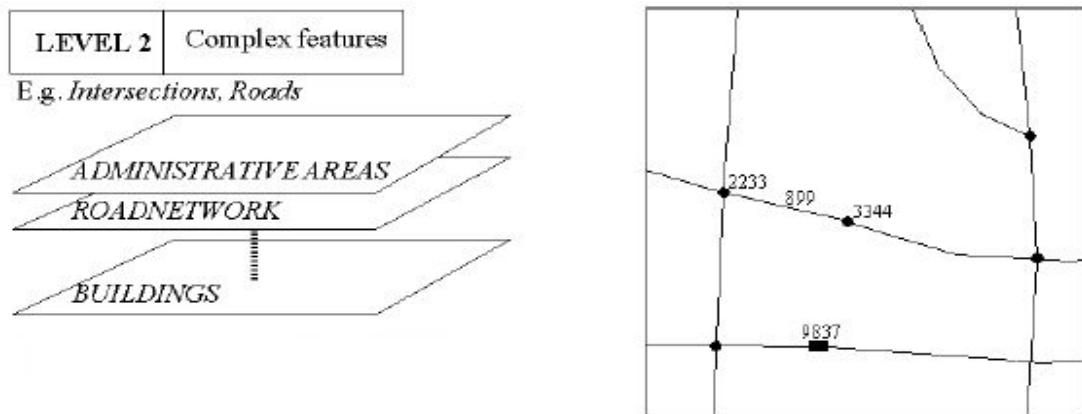
*Level 1: Features.* Level 1 is the most used level of GDF and contains simple features such as road elements, rivers, boundaries, and signposts. Features can have attributes that are specific (i.e. one way, road width, number of lanes) as well as relations that are very important for car navigation systems. Relations can be “forbidden turn from

road element 1 to road element 2” or “road element 1 has priority over road element 2”(Figure 4.6).



**Figure 4.6:** GDF Level 1, simple features

*Level 2: Complex Features.* At this level the “simple features” are aggregated to a higher-level feature. For instance, at Level 1 all road elements of an intersection should be represented, but at Level 2, the intersection is only represented with a single point. Level 2 is mostly used when a simplified description of the road network is sufficient. For instance, inter urban route calculation does not require a high level of detail, but vehicle location by means of a GPS receiver does need the detailed description of the road network (Figure 4.7).



**Figure 4.7:** GDF Level 2, complex features

#### 4.2.2. Road database

The road database tables in CIM-TR are created according to GDF specifications [24]. with some variations. The main tables and their fields that are used for different levels are given in Tables 4.4,4.5 and 4.6.

**Table 4.4:** Main tables from database for Level 0

Edges	Nodes	XY_coordinates
Edge_ID	Node_ID	XY_ID
XY_ID	XY_ID	latitude
From_node_ID	Face_ID	longitude
To_node_ID		
Left_face_ID		
Right_face_ID		

**Table 4.5:** Main tables from database for Level 1

Lines	Points
Line_ID	Point_ID
Edge_ID	Node_ID
From_point_ID	Point_name
To_point_ID	
Line_name	
Line_width	

**Table 4.6:** *Complex\_Features* table from database for Level 2

Complex_Features
Complex_feature_ID
Feature_ID
From_feature_ID
To_fearture_ID
Complex_feature_name

The hardest part was to fill the *XY\_coordinates* table. To give an idea, there are 1081 coordinate records to define the road elements for E5 from Tuzla Tersane to Avcilar and Avcilar to Tuzla Tersane. There were two options to fill this table: to go and stop at every one of these points and send a message to the server or get these coordinates

from Google Earth. The coordinates taken from Google Earth was accurate enough that second option was chosen.

There is a detailed procedure to decide if a location is on a road preloaded to the database. Two new tables are used for these procedures: *User\_last\_location* and *Road\_speed* (Table 4.7). *User\_last\_location* keeps the last location of the mobile if at the end of the procedures mobile seems to be on one road or on two roads. The latter case is possible on two sided roads when the user drives close to opposite side. If mobile seems to be on one road the road is recorded as the *CertainRoad*, whereas if it seems to be on two roads, the two roads are recorded as the two *Candidate Roads*.

**Table 4.7:** User\_last\_location and Road\_speed tables

User_last_location	Road_speed
IMEI	Road_speed_ID
Last_date	Date
Last_time_ID	Time
Last_latitude_ID	Road_ID
Last_longitude	
Cand_roadEle_ID1	
Cand_roadEle_ID2	
Cert_road_ID	



```

int IfOnRoad(double latitude, double longitude, String IMEI, Calendar cal)
{
    CHECK IF THE DEVICE SENT IN X TIME
    long[] CertCandRoad_ID = IfDeviceSentInXTime (xSecs, latitude, longitude, IMEI, cal);

    IF IT HAS SENT

        IF THERE IS A CERTAIN ROAD
            GET THE DETAILS OF THE ROAD

                IF IT IS ON THE SAME LINE
                    IfOnTheLine(latitude, longitude, line_width, fr_point_latitude,
                    fr_point_longitude, to_point_latitude, to_point_longitude );
                    FIND THE ROAD THE SEGMENT BELONGING TO
                    UPDATE User_last_Location AND Road_Speed tables

                ELSE BEHAVE THE coordinate as a new point
                    FirstLocationInfo(latitude, longitude, IMEI, cal);

            ELSE IF IT IS ON TWO CANDIDATE ROADS CALL THE TwoCandidateRoads
                TwoCandidateRoads(IMEI, last_latitude, last_longitude, latitude, longitude,
                CertCandRoad_ID[2], CertCandRoad_ID[3], cal);

        ELSE // IT HAS NOT SENT
            FirstLocationInfo(latitude, longitude, IMEI, cal);

    return 0;
}

```

**Figure 4.8:** *IfOnRoad* algorithm

### 4.2.3. Locating Algorithms

When application receives a new location message, after decomposing the messages it *IfOnRoad* main function (Figure 4.8). It checks if the same device sent data in X seconds. X is a variable and default it is taken as 20 since the vehicle can probably change the road segment it is on after 20 seconds. If it has sent, it checks if the mobile was on *CertainRoad* or two *Candidate Roads*. This saves time. If it is on *CertainRoad*, then it is checked if it is on the same road. Otherwise the location is treated as first data from user. If it is on two *Candidate Roads*, old location data is taken into calculations along with new location to decide on the direction of the mobile and locate the mobile on one *CertainRoad*. *IfOnTheLine* procedure is very similar to *IfOnTheLineWithDirection* except that it does not receive and take into

calculations the old location data. *IfOnTheLineWithDirection* algorithm is given in Figure 4.10 .

```
int FirstLocationInfo (double latitude, double longitude, java.lang.String IMEI, Calendar cal)
{
GET THE POINTS WHICH ARE THE STARTING POINTS OF THE LINES WHERE ,
POINTS ARE IN THE SQUARE WITH max_distance AS A RADIUS WITH
LOCATION coordinates, (latitude AND longitude) AT THE CENTER

    AS LONG AS THERE ARE POINTS

        CHECK IF THE coordinates IS ON THE LINE WITH POINT IS THE
        STARTING POINT IF IT IS NOT CHECKED BEFORE
        IfOnTheLine(latitude, longitude, line_width, fr_point_latitude,
fr_point_longitude, to_point_latitude, to_point_longitude );

        CHECK IF THE coordinates IS ON THE LINE WITH POINT IS THE END
        POINT IF IT IS NOT CHECKED BEFORE
        IfOnTheLine(latitude, longitude, line_width, fr_point_latitude,
fr_point_longitude, to_point_latitude, to_point_longitude );

    IF THE coordinates SEEMS TO BE ON ONLY ONE LINE
        FIND THE ROAD THAT THE SEGMENT BELONGING TO
        UPDATE User_Last_Location(CERTAIN_ROAD) and Road_Speed

    IF THE coordinates SEEMS TO BE ON TWO ROADS
        UPDATE User_Last_Location(CAND_ROAD1, CAND_ROAD2);

    return 0;
}
```

**Figure 4.9:** *FirstLocationInfo* algorithm

```

int IfOnTheLineWithDirection (double ex_latitude, double ex_longitude,
double latitude, double longitude, Integer line_width, double fr_point_latitude, double
fr_point_longitude, double to_point_latitude, double to_point_longitude)
{
    FIND THE EQUATION OF THE LINE PASSING THROUGH TWO POINTS
    A = (y2 - y1);
    B = (x1 - x2);
    C = (y1*(x2-x1)+x1*(y1-y2));

    FIND THE distance FROM coordinates TO THE LINE

    distance = Math.abs(A*x0 + B*y0 + C) / Math.sqrt(A*A + B*B);

    FIND THE CLOSEST POINT OF THE LINE TO THE coordinates

    IF THIS POINT IS ON THE LINE SEGMENT
        CALCULATE dis_extostart WHICH IS DISTANCE FROM OLD coordinates TO
        THE STARTING POINT

        CALCULATE dis_itostart WHICH IS DISTANCE FROM NEW coordinates TO
        THE STARTING POINT

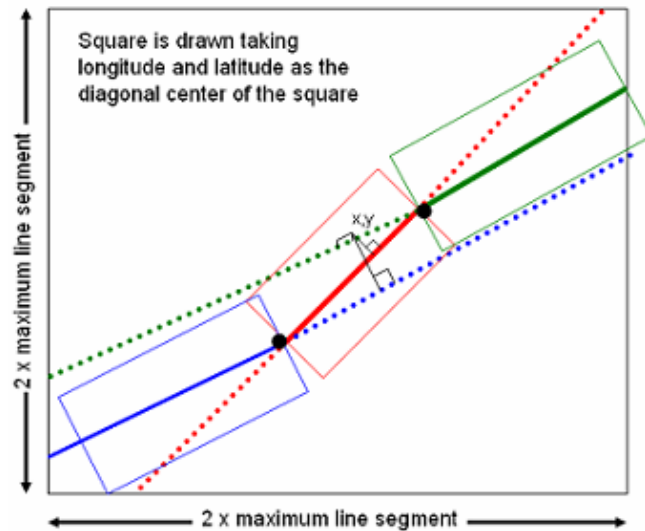
        IF THE distance IS SMALLER THAN THE line width
            IF dis_itostart IS GREATER THAN dis_extostart
                THE coordinates IS ON THE LINE, AND IT HAS MOVED IN ROAD
                DIRECTION return 2
            ELSE /
                THE coordinates IS ON THE LINE, AND IT HAS MOVED OPPOSITE TO
                ROAD DIRECTION return 1

        ELSE
            THE coordinates IS NOT ON THE LINE, return 0
    ELSE
        THE coordinates IS NOT ON THE LINE, return 0
}

```

**Figure 4.10:** *IfOnTheLineWithDirection* algorithm

If the client application has not sent any message within a preset period of time recently or the device is not on the *CertainRoad* or the two *Candidate Roads*, the received location information is treated as the first data coming from that particular mobile device (Figure 4.9). A virtual square is drawn with an edge twice as long as the maximum line segment where the received location coordinate stays at the center of the square. It is checked whether the location is on the lines which have any part inside this square (Figure 4.11).



**Figure 4.11:** *FirstLocationInfo* and *IfOnTheLine* Procedures Illustration

Depending on the result it can be declared as moving along on the *CertainRoad* or the two *Candidate Roads*. In these procedures if the location comes out to be on a *CertainRoad*, new speed information is recorded with a timestamp in the *Road\_speed* table.

```

int TwoCandidateRoads (String IMEI, double last_latitude, double last_longitude, double
latitude, double longitude, long CandRoad_ID1, long CandRoad_ID2, Calendar cal)
{
CHECK IF coordinates IS STILL ON CAR1 AND IF YES MOVED ON ROAD DIRECTION
SET CAR1_RD and CAR1_NotRD
IfOnTheLineWithDirection (last_latitude, last_longitude, latitude, longitude, line_width,
fr_point_latitude, fr_point_longitude, to_point_latitude, to_point_longitude);

CHECK IF coordinates IS ON CAR1_Next WHICH IS THE ROAD JUST AFTER ROAD CAR1
SET CAR1_Next
IfOnTheLine (latitude, longitude, line_width, fr_point_latitude, fr_point_longitude,
to_point_latitude, to_point_longitude);

CHECK IF coordinates IS ON CAR2 AND IF YES MOVED ON ROAD DIRECTION
SET CAR2_RD and CAR2_NotRD
IfOnTheLineWithDirection (last_latitude, last_longitude, latitude, longitude, line_width,
fr_point_latitude, fr_point_longitude, to_point_latitude, to_point_longitude);

CHECK IF coordinates IS ON CAR2_Next WHICH IS THE ROAD JUST AFTER ROAD CAR2
SET CAR2_Next
IfOnTheLine (latitude, longitude, line_width, fr_point_latitude, fr_point_longitude,
to_point_latitude, to_point_longitude);

TAKE THE ACTIONS GIVEN IN THE TABLE ACCORDING TO RESULTS
FIND THE ROAD THAT THE SEGMENT BELONGS TO
UPDATE BOTH User_Last_Location AND Road_Speed FOR CERTAIN ROAD
FIND THE ROADS THAT THE SEGMENTS BELONGS TO
UPDATE User_Last_Location FOR CANDIDATE ROADS
}

```

**Figure 4.12:** *TwoCandidateRoads* algorithm

If *TwoCandidateRoads* function (Figure 4.12) is called, then it means there is a new coordinate and two candidate roads. First it is checked if the coordinate is on the first candidate, if yes if it is moved on road direction or moved in opposite direction. This is done via *IfOnTheLineWithDirecton* function which is given in Figure 4.10. In this procedure old location data is taken into calculations along with new location to decide on the direction of the mobile. There is another possibility that the coordinate can be on the next road. Three parameters are set accordingly. The same procedure is run for second candidate. According to these six parameters, actions, those are given in Table 4.8, taken. The possibilities that are not given in the table are not possible in real life but in case they occur then the coordinate is neglected.

**Table 4.8:** Actions to take in *TwoCadidateRoads*

CAR1_RD	CAR1_NotRD	CAR1_Next	CAR2_RD	CAR2_NotRD	CAR2_Next	
T	F	F	F	F	F	SET CAR1 AS Certain Road
F	F	F	T	F	F	SET CAR2 AS Certain Road
F	F	T	F	F	F	SET CAR1_Next AS Certain Road
F	F	F	F	F	T	SET CAR2_Next AS Certain Road
T	F	F	T	F	F	KEEP CAR1 AND CAR2 Candidates
F	T	F	F	T	F	CALL <b>FirstLocationInfo()</b>
F	F	T	T	F	F	SET CAR1_Next AND CAR2 AS Candidates
T	F	F	F	F	T	SET CAR1 AND CAR2_Next AS Candidates
F	F	T	F	F	T	SET CAR1_Next AND CAR2_Next AS Candidates
T	F	F	F	T	F	SET CAR1 AS Certain Road
F	T	F	T	F	F	SET CAR2 AS Certain Road
F	F	T	F	T	F	SET CAR1_Next AS Certain Road
F	T	F	F	F	T	SET CAR2_Next AS Certain Road

Single thread of *Server* application can handle all the messages sent by the different mobiles in the tests. If the contributor number increases so much, the *IfOnRoad* and its sub procedures can be called as a different thread for each message.

### 4.3 Communication

There are three options to connect a MIDP application to Web Services. If the mobile device supports Java JSR 172 API then there can a two tier SOAP communication be established easily. If not then are two options left: Using kSOAP and kXML or establishing a three tier Web Service Client.

JSR-172 is designed to provide an infrastructure as two optional packages for J2ME to:

- provide basic XML processing capabilities
- enable reuse of web service concepts when designing J2ME clients to enterprise services
- provide APIs and conventions for programming J2ME clients of enterprise services
- adhere to web service standards and conventions around which the web services and Java developer community is consolidating
- enable interoperability of J2ME clients with web services
- provide a programming model for J2ME client communication with web services, consistent with that for other Java clients such as J2SE

There is one disadvantage that JSR-172 is available only on new mobile phone models. Our client were established on the phone Siemens SXG75 and during researches it was seen on the Siemens developer site that the product does not support JSR-172. But Siemens SXG75 was the first mobile phone to support GPS, newer phone models with GPS receiver also have support for JSR 172.

KSOAP and KXML are two open source packages designed to enable SOAP and XML applications to run within a KVM which is a low-memory virtual machine running on a mobile phones. They are thin, easy to use, and well documented. Combined into a single jar file, they take up less than 42K. KSOAP with Beta version was released in May 2001 by Enhydra organization. Version 1.2 used for a long time which was announced in 2002 and in 2006 now there is version 2.1.1 released.

There was the option to go with KSOAP or three tier Web Service client. We made some trials with KSOAP and saw that it is still a developing platform and need extra effort to have successful communication. So the three tier web service client mechanism offered by Java [26] is used .

This type of connection utilizes three "tiers": client, web application containing servlet, and server. The client communicates with the generated middleware servlet using a proprietary communication protocol. The servlet and server communicate using standard Simple Object Access Protocol (SOAP) messages.

The code generated by the Mobile Client to Web Application Generator has a footprint that can be as small as 3 kilobytes, so it can be deployed on low-end Java ME MIDP devices. The network bandwidth used in calls from the client to the server is also efficient, typically using only slightly more than the minimum needed to transfer the raw data.

The Mobile Client to Web Application Generator creates:

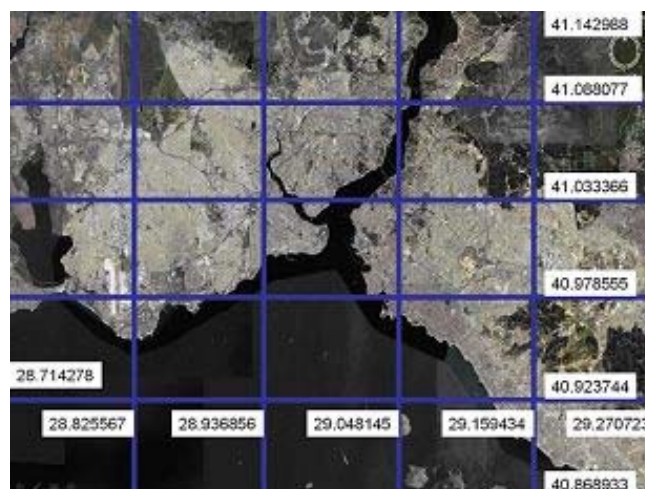
- A Java ME client class
- A servlet and supporting classes
- A mapping file in xml format that can be used to regenerate the server and Java ME client classes.
- Optionally, a MIDlet you can examine and modify.

## 4.4 Traffic Calculation and Display Application

*Display* application is a Java applet that connects to the database to retrieve user locations and speed information and display it to the public. It has two main menu items in its menu bar: *Who is online* and *Traffic Info*. *Who is online* shows the locations and speeds of mobile phones that has sent data in last 1 hour on a map. But with just small changes it can display any recorded mobile phone for chosen period of time for tracking. This can be helpful if we would like to offer free tracking system to the contributors. Companies can track their vehicles on our system free while their vehicles are producing traffic data for this project. We won't display their names but rather display contributor IDs. If they register to the system we can display their nicknames if they would request in their registration.

### 4.4.1. Who is online?

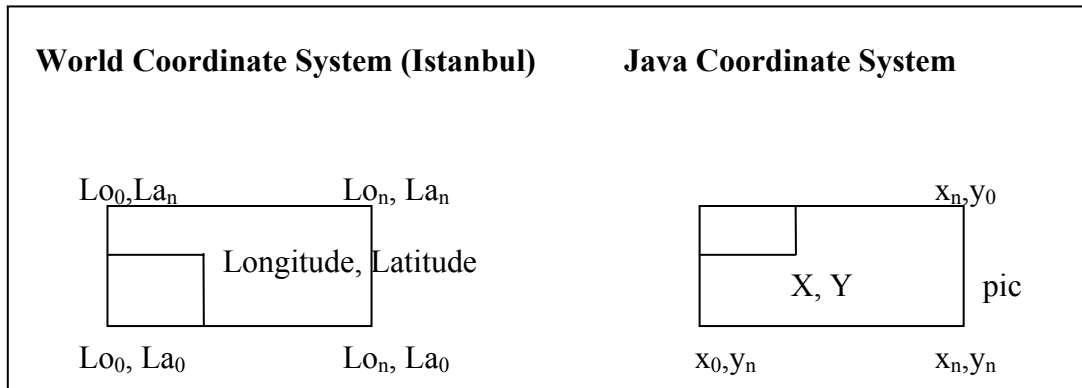
When *Who is online* menu item is clicked a `JInternalFrame` is created. Istanbul Map is divided into  $5 \times 5 = 25$  equal parts and each part is inserted onto `JInternalFrame` as `JPanel` in the correct position. The maps start from 29.270723E Pendik and ends on 28.714278E Kucukcekmece on East-West direction. on South-North direction, map starts at 40.868933N and ends at 41.142988N. Every map component is a rectangle with  $0.111289^\circ \times 0.054811^\circ$  dimensions. The limits of these maps are set as parameters to extend the map and cover other areas. This map covers the city of Istanbul (Figure 4.13).



**Figure 4.13:** Istanbul map is divided into  $5 \times 5 = 25$  equal components



*Who is online* shows mobile locations by projecting World Coordinate System on a picture using Java coordinate system. So it is very easy to cover other cities and even highways between cities. The general mechanism used in *Who is online* to show the coordinates on a map is as follows:



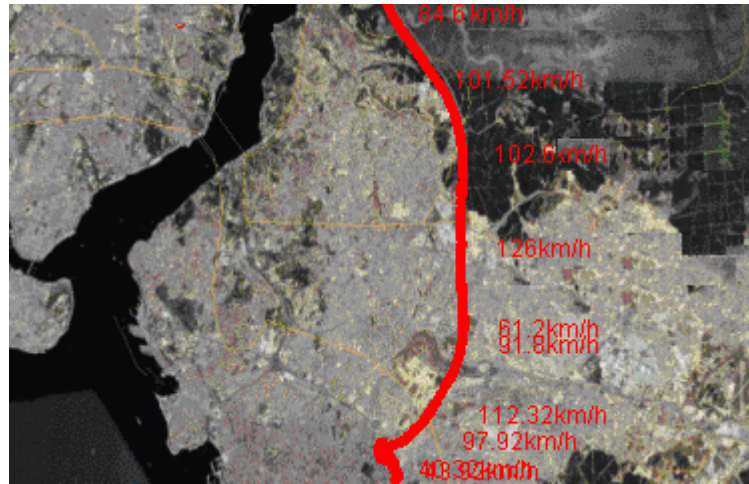
**Figure 4.14:** World Coordinate System vs Java Coordinate System

$$X = (\text{Longitude} - L_{o_0}) / ((L_n - L_{o_0}) / (x_n - x_0)) \quad (4.1)$$

$$Y = -(\text{Latitude} - L_{a_n}) / ((L_{a_n} - L_{a_0}) / (y_n - y_0)) \quad (4.2)$$

where  $(L_{o_0}, L_{a_0})$  and  $(L_n, L_n)$  are the minimum and maximum coordinates of the covered area respectively. And on Java Coordinate System  $(x_0, y_0) = 0,0$  and  $(x_n, y_n) = n,n$  (Figure 4.13) By these calculations we find the value of X,Y on the Java coordinate system according to longitude and latitude value of a location on World Coordinate System.

Application gets the location and speed information one by one from the *Location* table in the database for last 1 hour, checks in which map component limits the location is, and draws a small oval on the corresponding point on the corresponding JPanel. It also displays one speed data of the mobile for every 40 location data. When one of the JPanels is clicked that JPanel fits into the whole JFrame and a better resolution of that part of the map is loaded at the screen of the Java applet. According to implementation the applet displays the locations and speeds reported in last 1 hour with the vehicles speed (Figure 4.15).



**Figure 4.15:** Part of *Who is online* application screen

#### 4.4.2. Traffic info

The second menu item in the *Display* application user interface is *Traffic Info*. *Traffic Info* will have the city names as the sub menu items. Right now there are only roads from Istanbul in the database therefore the only submenu is Istanbul for now.

When Istanbul is clicked on the Traffic Info menu item, a *JInternalFrame* is created. The Istanbul map is divided into  $5 \times 5 = 25$  equal components fit in *JPanels* which can be clicked on to cover the whole *JInternalFrame* like *Who is online*.

One of the important part of the CIM-TR project was to decide how to calculate the traffic on a road especially if there is no speed data at current date and time. A decision has to be made when there is no current speed data for a particular road. If there is a feed in about a particular road within less that half an hour it may be accepted and published as the current data. However the use of statistical data becomes essential if there is no such data available.

It was decided to categorize statistical data and use them with different coefficients when calculating the traffic information. Five different set of speed information are returned from database starting from the least valuable to most valuable one. These sets are listed below.

- 1) Average of speeds everyday in that hour(i.e.16:49-17:49)
- 2) Average of speeds in that weekday in that hour(i.e.Thursdays 16:49-17:49)
- 3) Average of speeds in one hour(i.e.Today 16:49-17:49)
- 4) Average of speeds in half an hour(i.e.Today 17:19-17:49)
- 5) Average of speeds in 10 minutes(i.e.Today 17:39-17:49)

The highest number in the order indicates the dominance of the figure that we have to consider. Moreover the data in the lower numbered average includes the average of the higher ones. In another point of view if there is no available data for a specific level, there is no data for higher levels. Since data in set 5 is considered as instant, if there is data available in set 5 then it contributes 80% to the result and the rest 20%. Otherwise higher level sets contribute 60% whereas lowers contribute 40% totally. The contribution of sets according to the availability of data in sets given in Table 4.9.

**Table 4.9 : Percentage contribution of sets to the result**

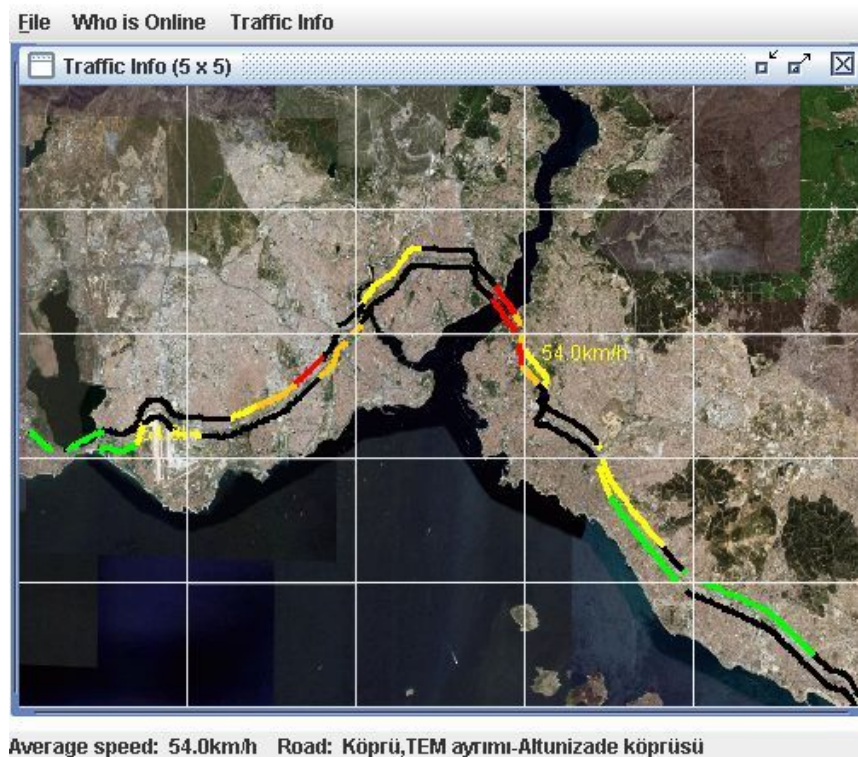
Data in set(s)	1	1,2	1,2,3	1,2,3,4	1,2,3,4,5
1	100	40	16	6,4	1,28
2		60	24	9,6	1,92
3			60	24	4,8
4				60	12
5					80

The roads are displayed in colors that are set according to the speed information returned from the database (Table 4.10).

**Table 4.10: Display colors for the road traffic**

Traffic condition	Speed(km/h)	Color
No information		Black
Very heavy traffic	0-30	Red
Heavy traffic	30-50	Orange
Running Traffic	50-80	Yellow
Open traffic	80-	Green

*Traffic Info* calculates the average speed of each road that is in the *Complex\_Features* table and defines display colors according to these speeds. It gets the starting and end point coordinates of each road element that are in the *Line* table and draws lines from starting points to the end points for each road element in display colors of each road. This drawing again uses the projection of World Coordinate System on Java Coordinate System. Mouse cursor can be moved on the road to see the name of the road and exact average speed on that road (Figure 4.16). So there are two mechanisms for mouse. When mouse is clicked on a part of the map that part covers the *JInternalFrame*, moreover when mouse cursor is moved on the road the speed and name is displayed on the status bar. The coordinates of the two sided roads are very close to each other; therefore a mechanism is also established to set a distance between opposite roads.



**Figure 4.16:** *Display* application interface

The User Interface of *Display* Application is given in 4.16.

## 5. EXPERIMENTAL RESULTS

There were three experiments carried out to test the CIM-TR application. In the first experiment a Tester application is produced to test the system stability against various numbers of contributors sending messages with different parameters for long period of time. In the second and third experiments, clients on mobile phones are used to test the algorithms produced to locate the locations on roads and to calculate traffic, respectively.

### 5.1. Experiment 1

When the system is tested by limited number of mobile phones for limited number of times, it works fine. But it is important to see the system stability when continuous and various number of client devices contribute to the project. A *Tester* application is created that can simulate mobile devices traveling at different speeds on the roads in different hours of the day. The default values and variables of the Tester application is set according to observations on the traffic and CIM-TR system with real data.

The traffic in 24 hours of a day is divided into 5 different categories. These categories and the default values are given below:

1) Hours between: 00:00 – 07:00 : It is night and there is open traffic on every road. One mobile device starts the trip in every hour in each direction of each main highway.

Average speed: 24 m/s = 86.4 km/h

2) Hours between 07:00 – 09:00: There is a heavy traffic since it is the start of the business hours. One mobile device starts the trip in every 20 minutes in each direction.

Average speed: 6 m/s = 21.6 km/h

**3)** Hours 09:00 – 17:00: It is the middle of the day. There can be running or heavy traffic on the roads. One mobile device starts the trip in every 30 minutes in each direction.

Average speed:  $14 \text{ m/s} = 50.4 \text{ km/h}$

**4)** Hours 17:00 – 19:00: The traffic behavior is very similar to group 2. It is time for people to go home from work. One mobile device starts the trip in every 20 minutes in each direction.

Average speed :  $6 \text{ m/s} = 21.6 \text{ km/h}$

**5)** Hours 19:00 – 00:00: It is the evening hours. Most people are at home. There can be running or heavy traffic on the roads. One mobile device starts the trip in every 30 minutes in each direction.

Average speed :  $14 \text{ m/s} = 50.4 \text{ km/h}$

Each device starts from one end in east - west and west-east direction. The coordinates that they will follow are kept in files. An application *Route Creator* produces the routes that devices follow. The *Route Creator* application connects to road database and reads the start and end coordinates of each road. Then it calculates and writes the middle point of each road to separate files for each direction.

The *Tester* application creates one thread for each mobile device according to the rules given above. It passes four parameters to the threads:

**1)** *routeID*: defines the routes devices track

**2)** *counter*: defines the end of the IMEI number which shall be unique for each travel.

**3)** *TBL*: Time Between Locations variable that is sent on *header* parameter of messages (Figure 4.2).

**4)** *NOL*: Number Of Locations that are sent in each message (Figure 4.3).

Each thread creates a random *CarDriverVariable* variable which can have an effect of 50% to the speed value. In every *TBL* seconds the thread checks the time, set a default speed value *Speed* according to the traffic categories listed above. Then it creates another random variable *ChanceVariable* in that second which can have 25% effect to the speed value. The final Speed value at that time is calculated with the formula given below where *CarDriverVariable* can take values between -50 to 50 and *ChanceVariable* can take values between -25 to 25.

$$Speed = Speed + Speed * CarDriverVariable / 100 + Speed * ChanceVariable / 100 \quad (4.3)$$

The thread reads the latitude and longitude from *Route* file and adds to the *Location\_data* parameter. In every *TBL\*NOL* seconds, it prepares the *header* and sends the message to the server. The thread stays alive as long as there are coordinates on the *Route* file.

Tester application ran 24 hours in experiment 1. There were three different *TBL* and *NOL* combinations used. Server Application successfully handled 48236 location data coming from 90 different mobile phones via 12958 messages (Table 5.1). The success rate was very high as expected on locating the road elements the mobiles were moving since routes are produced from the road database. But this experiment was done to see the system stability against various numbers of contributors with different *TBL* and *NOL* parameters sending messages for long period of time.

**Table 5.1:** Number of locations handled by *Server* Application in 24 hours

<b>TBL(sec)</b>	<b>NOL</b>	<b>Mobiles</b>	<b>Messages</b>	<b>Locations</b>
5	5	52	5564	27820
8	2	24	6456	12912
2	8	14	938	7504
Total		90	12958	48236

## 5.2. Experiment 2

In experiment 2, the aim is to find the success rate of the algorithm used to locate the road that the location information has been sent from. This is calculated by

finding the rate of records that was entered to the *road\_speed* table to the total number of locations sent by test mobile in certain part of highway in certain amount of time. The road is chosen as E5 between Kartal and Erenköy because of practical reasons which is approximately 12 km and have different number of lanes such as 4 around Küçükyalı and 2 around Cevizli.

At the end of the test runs, the *Server* application procedure that was used to find the road that the location information has been sent from came out with a success rate of 91% (Table 5.2). This result is very nice but when we look at the loss there can be three main reasons. As it can be seen from Figure 4.11, procedure treats roads as aligned rectangles and the connection points of these rectangles causes small uncovered areas. Second reason is that road widths are kept as lane numbers and Istanbul roads do not have the standard lane widths. Of course one of the important error factors in location based services is the sensitivity of the GPS receivers.

**Table 5.2:** Success rate of road locating procedure

	<b>First run</b>	<b>Second run</b>	<b>Third run</b>	<b>Total</b>
<b>Records in <i>road_speed</i></b>	234	251	214	699
<b>Records in <i>location</i></b>	252	286	233	771

### 5.3. Experiment 3

When there are several users on roads, the CIM-TR will supply sufficiently correct traffic data to the user but when there are not online users on a road, displaying information using statistical data is a hard job. The mechanism that was implemented in the CIM-TR gives better results as long as there are data in most valuable sets mentioned before. Otherwise there can be some differences between the actual speed and the calculated one. But in general this difference is not much since the coefficients for different sets are chosen according to observations in Istanbul traffic and database. Table 5.3 gives an example from database records for a road and the results that would be displayed according to the sets taken into calculations. The values for displayed results are calculated with that set and the ones before. The



instant speed value was 21.0 in this test and as it can be seen from table the results displayed gets closer to the instant value as there is data in more valuable sets.

**Table 5.3:** Displayed results according to data sets

<b>Results</b>	<b>Set 1</b>	<b>Set 2</b>	<b>Set 3</b>	<b>Set 4</b>	<b>Set 5</b>
<b>Returned</b>	15,7	17,2	20,2	22,3	21,1
<b>Display</b>	15,7	16,6	18,8	20,9	21,1

## 6. CONCLUSION

The IMOGA which is developed to collect data from ubiquitous mobile devices and use in Grid applications, has been achieved its goal setting up the architecture and on collecting location and speed data from mobile devices with integrated GPS receiver via CIM-TR application. This data is the most valuable data we can get from mobile devices without developing an external accessory for now. Mobile devices with the CIM-TR client installed can send location and speed information in SOAP based XML messages to the web service offered by the server application running on interlocutors. The communication mechanism developed, enables to pass data with minimum cost.

The coordinates of main highways in Istanbul are entered to the database. Traffic information on these highways is presented to the users on a map in a public web page opened to the internet. Visitors can see the average speeds on the roads separated by junctions in every 3 - 4 km, on an overall city map or more detailed maps for districts of the city. For now the information given in the WAP pages are not visual but just textual.

Increasing the number of mobile devices that runs client application is the most important fact for the CIM-TR to work with maximum correctness. First thing to do is to convince users to participate in a public project for their and others benefit. Then the IMOGA must offer other services than traffic information that would help to convince users. Right now vehicle tracking system is free for the participants. They can register to the system with their IMEI numbers and track their vehicles on map, live. Since users have to pay for their GPRS cost to the GSM operators, it is possible to come to an agreement with operators to make discount for the participants, reward for presenting the CIM-TR traffic information in their WAP sites. There are researches on triangular economic models between users, grid infrastructures and Internet service Providers (ISP) [9].

## **7. FUTURE WORK**

There are several areas that can the IMOGA expand. The most important implementation to be added to the IMOGA is to develop the mechanism that would connect server in interlocutor with Grid middlewares and share resources from ubiquitous mobile devices with Grid infrastructures. The communication between minions and interlocutor can go further that minions declare available services and data they can offer and interlocutor can keep an Universal Description, Discovery, and Integration (UDDI) directory for these services. It can check which minions are available in time intervals. Temperature, health monitoring and pollution sensors can be connected to the mobile phones as accessories via I<sup>2</sup>C wired and Bluetooth® wireless technologies. I<sup>2</sup>C communication mechanism with mobiles is worked on for the IMOGA recently. Implementing health monitoring sensors and using in Grid applications is a very promising research area. A European digital map company Automotive Navigation Data which is following IMOGA over the internet page declared that integrating AND data and technologies to the IMOGA project can open a commercial market in Europe for location based services in tracking, traffic management and health (Appendix).

## REFERENCES

- [1] **F.Berman, G. Fox, T. Hey**, 2003. The Evolution of the Grid in *Grid Computing - Making the Global Infrastructure a Reality*, pp. 65-100. **D.D. Roure, M.A. Baker, N.R. Jennings, N.R. Shadbolt**, John Wiley and Sons Ltd.
- [2] **L.McKnight, J.Howison, S.Bradner**, 2004. Wireless Grids Distributed Resource Sharing by Mobile, Nomadic, and Fixed Devices, *IEEE Internet Computing*.
- [3] **M.Tuisku**, 2004. Wireless Java-enabled MIDP Devices as Peers in Grid Infrastructure, First European Across Grids Conference, Santiago de Compostela, Spain
- [4] Gridblocks project page <http://gridblocks.hip.fi> (as of 04/05/2007)
- [5] Akogrimo project page <http://www.mobilegrids.org> (as of 04/05/2007)
- [6] Gridlab project page <http://www.gridlab.org> (as of 04/05/2007)
- [7] **I. Foster C. Kesselman**, 2004. Grid 2, Elsevier Inc.
- [8] **I. Foster, C. Kesselman, S. Tuecke**, 2001. The Anatomy of the Grid", Intl J. Supercomputer Applications.
- [9] **T.Phan, L.Huang, C.Dulan**, 2002. Challenge: Integrating Mobile Wireless Devices Into the Computational Grid, *MOBICOM' 02*, Atlanta Georgia, USA.
- [10] **L.McKnight, J.Howison**, 2003. Towards a Sharing Protocol for Wireless Grids, CCCT '03, Orlando, Florida, USA.
- [11] **J.Hwang, P.Aravamudham**, 2003. Proxy-based Middleware Services for Peer-to-Peer Computing in Virtually Clustered Wireless Grid Networks, CCCT '03, Orlando, Florida, USA.
- [13] **P. Grabowski, B. Lewandowski**, 2003. Mobile-enabled grid middleware and/or grid gateways, Grid Lab Project.  
<http://www.gridlab.org/Resources/Deliverables/D12.2.pdf>  
(as of 04/05/2007)

- [14] **N. Apte, K. Deutsch, R. Jain**, 2005. Wireless SOAP: Optimizations for Mobile Wireless Web Services. 14th World Wide Web conference.
- [15] **A. Hampshire**, Extending the Open Grid Services Architecture to Intermittently Available Wireless Networks  
<http://www.allhands.org.uk/2004/proceedings/papers/232.pdf> (as of 04/05/2007)
- [16] **C.Selman**. 2007, BenQ-Siemens Product Manager, Personal Interview
- [17] **Greg Easson, Lance Yarbrough**, 2001. GPS and GIS The University of Mississippi Geoinformatics Center.  
[http://umgc.olemiss.edu/pdf/workshop/umgc\\_gps\\_gis.pdf](http://umgc.olemiss.edu/pdf/workshop/umgc_gps_gis.pdf)  
(as of 04/05/2007)
- [18] **M. Kahveci, F. Yildiz**, 2005. GPS Teori Uygulama (GPS Theory and Applications), Nobel Publishing.
- [19] **L.C. Larijani**, 1998. GPS for everyone, American Interface Corp. New York.
- [20] **L. Srivastava, R. Kirvan, I.Silver**, 2006. The Regulatory Environment For Future Mobile Multimedia Services, International Telecommunication Union New Initiatives Workshop. Mainz Germany,
- [21] I2C specification, 2000.  
<http://www.standardics.nxp.com/literature/books/i2c/pdf/i2c.bus.specification.pdf>
- [22] Geographic Data Files Specification 3.0, 1996.  
[http://www.ertico.com/en/links/links/gdf\\_geographic\\_data\\_files.htm](http://www.ertico.com/en/links/links/gdf_geographic_data_files.htm)
- [23] **K.Holtman, B.Hertzberger, A.Hoekstra, R.Williams**, HEPGRID2001 : A model of a virtual data grid application, HPCN Europe 2001, Amsterdam, Netherlands.
- [24] Standard for Digital Geospatial Metadata 2.0, 1998.  
[http://www.fgdc.gov/standards/projects/FGDC-standards-projects/metadata/base-metadata/v2\\_0698.pdf](http://www.fgdc.gov/standards/projects/FGDC-standards-projects/metadata/base-metadata/v2_0698.pdf)
- [25] **ISO 19110**, 2005. Geographic information -- Methodology for feature cataloguing <http://www.iso.org> (as of 04/05/2007)
- [26] Java web page <http://java.sun.com> (as of 04/05/2007)

[27] **M. J. Yuan**, 2005. Enterprise J2ME: Developing Mobile Java Applications, Prentice Hall,

[28] **K. Topley**, 2002. J2ME in a Nutshell, O'Reilly and Associates Inc.

## APPENDIX: DECLARATION OF AND ABOUT IMOGA

### Automotive+Navigation Data

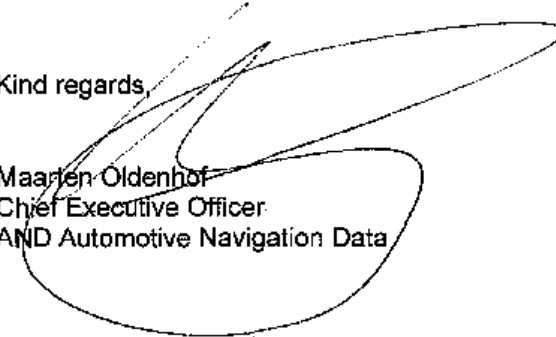
AND Automotive Navigation Data  
Van Vollenhovenstraat 3  
3016 BE Rotterdam  
The Netherlands

Tel: + 31 (0) 10 885 1200  
Fax: + 31 (0) 10 885 1240

Rotterdam: 30.03.2007

Integration of AND data and technologies to the IMOGA project can open a commercial market in Europe for location based services in tracking, traffic management and health. We are looking forward to work in this project as soon as Mobitech establishes commercial products and solutions.

Kind regards,



Maarten Oldenhof  
Chief Executive Officer  
AND Automotive Navigation Data

Attached

- 1 - IMOGA Project Description
- 2 - AND Company Information

# Automotive+Navigation Data

## ATTACHMENT 1

### **IMOGA (Integrating Mobile Devices in Grid Applications)**

#### **Project Description:**

Integrating mobile devices into Grid technologies and server applications can give ability to command power of supercomputers with a mobile device on one hand and can allow big applications to reach important data anywhere, anytime, on the other. IMOGA is planned to be an example to gather and share data that can be collected by ubiquitous mobile devices which can employ different kind of sensors such as Global Positioning System (GPS), temperature, health and pollution. In this project location and speed information that is produced by GPS enabled mobile devices such as mobile phones, is used. The developed client application running on mobile devices located in vehicles, such as the mobile phone of the driver, sends location and speed information to the server application in short time intervals via GPRS in the forms of Extended Mark-up Language (XML) like messages. The developed server application, which is preloaded with the highway coordinates via files in Geographic Data Files (GDF) format, locates the street that the vehicle is moving along and the received speed information is recorded along with a timestamp. A display application has also been implemented to calculate average of speeds that any vehicle may have at that very moment and post it on the Internet and WAP. If there is no actual data, i.e. there is no vehicle moving on a specific street, statistical data is utilised to produce such information. Thus foreseeing the traffic not only spatially but also in time is made possible. The client application running on the mobile device that feeds in data or any other devices that a client could name such as computers can exploit the information produced by the integrated system. Although producing and sharing traffic information is the essential aim of the project it is also shown that additional features such as vehicle tracking or even a systematic approach for traffic management can be done with IMOGA.



# Automotive | Navigation Data

## ATTACHMENT 2

### **Company Information: AND Automotive Navigation Data**

#### **AND Automotive Navigation Data**

AND Automotive Navigation Data is leading provider of digital mapping data used for location-based services around the world. AND Automotive Navigation Data focuses on the development of digital maps in Central and Eastern Europe, North-Africa, Central and South America and Australia. The digital maps are used in personal and 'in-car' navigation, Internet-based mapping, fleet management and more. The company employs approximately 250 employees. The company was founded in 1984, is headquartered in Rotterdam, the Netherlands and is listed on Euronext Amsterdam (AND).

'AND' as used represents AND Products B.V., AND International Publishers N.V. or AND Automotive Navigation Data.

#### **Precision navigable road maps by AND**

AND has collected worldwide road and address information for many years to create digital maps. These maps are used for navigation across the world by many consumers and companies. AND aims to collect data at a high detail level for new geographical areas such as Eastern Europe, North Africa, Asia Central and South America.

#### **Partnerships**

Google launched Africa map data in November 2006. The map data used is supplied by AND Automotive Navigation Data (AND). The AND map data for North Africa offer the highest coverage available in the market.

AND and Siemens VDO Trading GmbH have entered into an agreement for the integration of the AND Global Road Data for Eastern Europe, Africa and South America. The data will become part of the VDO Dayton in-car navigation systems through retrofitting.

Microsoft uses AND Mapping Data for Windows Live™ Local and MapPoint Web Service

## RESUME

**Name Surname** : Ersan Öztürk  
**Address** : Uğur Mumcu mah. Yunus  
Emre cad. 39/21  
Kartal/Istanbul/Turkey  
**Home phone** : +90 216 476 0113  
**Mobile phone** : +90 542 816 77 01  
**E-mail** : ersan6@yahoo.com  
**Birth date** : Feb.15, 1981  
**Sex** : Male  
**Nationality** : Turkish  
**Education level** : MSC student  
**CGPA Undergraduate:** 3.86 (1<sup>st</sup> in rank)  
**CGPA Graduate** : 3.50



### Education Info ;

2004- Istanbul Technical University-Computer Engin. MSC  
1999-2004 Yeditepe University - Computer Engineering  
1998-1999 The Barstow School (USA)  
1992-1998 Darüşşafaka Lycee (Turkey)

### Experience;

- *Partnership with BenQ APAC Turkey Office*  
(January 01 2007 - )  
-product management
- *Partnership with BenQ-Siemens Turkey*  
(October 01 2005-December 31 2006 )  
-product management
- *Partnership with SIEMENS Mobile Turkey*  
(September 01 2004- October 01 2005)  
-product management (6 months as a product manager while he was in military service)

- *Founder Partner of Mobitech Ltd. Şti.*  
(August 2004)
- *SIEMENS Mobile Turkey*  
(September 15 2003-September 01 2004)  
-my-siemens.com/turkey website management, product manager assistant
- *Yeditepe University Student Assistance*  
(February 01 2003-June 01 2004 )  
-in Assembly and Operating System laboratories.
- *SIEMENS e-business department*  
(August 15 2003-September 15 2003)  
-e-business basics, generic content management, statistics and interface usage.
- *IBM Turk. Dotcom department*  
(July 15 2003- August 15 2003)  
-IBM products, e-business basics, PHP and MySQL.
- *Yeditepe University Data Automaton Center*  
( July 01 2002-August 01 2002)  
-HTML, SQL and database concept.

**Computer Knowledge:**

C, C++, Java high level programming languages and assembly language as lower level, experienced in Motorola 6802. Also experience in SQL, worked on Oracle database and JDeveloper tool. Studied HTML, PHP, MySQL, XML and SOAP about internet technologies. Worked on Linux and made a project on Linux Kernel. Studied computer networks and made research on GPRS, UMTS and other mobile communication technologies. Recently working on mobile applications.

## **Foreign languages :**

English: Very good

## **Projects:**

### **Graduate Projects:**

- *Graduate thesis Subject: Mobile Applications and Grid*  
Undergraduate thesis Project: Integrating Mobile devices into Grid Applications
- *Subject: Expert System*  
Project: A Decision Support System for Mobile Phone Fault Diagnosis Using Expert System
- *Subject: Performance Analysis Of Computer Networks, using Network Simulator*  
Project: Performance Analysis on a simulation network.
- *Subject: Interconnection Protocols*  
Project: Java radio: Control of a radiochip with a basic Java application on PC using I<sup>2</sup>C
- *Subject: Parallel Algorithms*  
Project: Parallel implementation of SIMD Gauss Jordan algorithm

### **Undergraduate Projects:**

- *Undergraduate thesis Subject: Wireless Sensor and Ad-hoc networks*  
Undergraduate thesis Project: Evaluation and Testing of Sema Protocol on Mobile Adhoc Network Testbed
- *Subject: Computer Networks*  
Project: Reserach on GPRS and UMTS architecture and protocols
- *Subject: Artificial Intelligence*  
Project: Building up ISA and HASA hierarchy
- *Subject: Image Processing*  
Project: Extracting blood vessels from biomedical images
- *Subject: Database Management*  
Project: Producer-Seller-Consumer program using Jdeveloper
- *Subject: Software Engineering*  
Project : SOAP communication between e-business departments
- *Subject: Operating System and Design*  
Project : System Calls&Fair Scheduling in Linux Kernel
- *Subject: Digital Electronics*  
Programing EEPROM by communication via parallel port
- *Subject: Principels of programming languages-Java Language*  
Project: Mobile Network Simulation
- *Subject: Data Structure-C++ language*  
Project: MVP simulation; C++ language

**Hobbies ;**

Photography, soccer(high school team), basketball(high school team), track field, cultural travels, books.

Member of Darüşşafaka Community, AFS Community.

**References;**

Prof . Dr. Ahmet Dervişoglu - ahmetdervis@e-kolay.net

Prof. Dr. Sebnem Baydere - sbaydere@cse.yeditepe.edu.tr

Erem Karabey (Google Turkey –former director, Siemens Mobile

Turkey – former marketing manager) – 0533 283 9201

Ali Yılmaz (Siemens Mobile Turkey- former director) – 0533 324 6420