MASTER THESIS

2004-07-01

MONGER

A LOCATION DEPENDENT INFORMATION SYSTEM FOR USE IN TRAFFIC

By: Benjamin Özmen

boz@kth.se

http://room40.se/benjamin

KTH Microelectronics
and Information Technology

Abstract

In this report the design and development of a location dependent Traffic Information System, the Monger prototype is discussed.

Traffic Information Systems of today use FM-radio to distribute information to road users. This information is often only of interest to a small group of people that is travelling near the reported area, not all road users are interested of the information.

Monger is a Traffic Information System that uses peoples everyday motion to transport and distribute local information. Here we assume that the road users are the ones with greatest knowledge about the roads that they are travelling on and that they are willing to share information to fellow road users.

The Monger application is location dependent. Each message is appended with GPS data which specifies where the message has been posted and when it should be played. By using GPS data, only people that are within a defined area[1] will be aware of the current message left there. This message will not appear at the device of other users outside the specified area. Our application is based on a serverless solution without any assumption on the underlying topology and connectivity of the network.

Information distributed in this kind of system is mainly about obstacles or accidents along the roads, but can also be applied in other situations.

[1] Defined area: The area in the vicinity of where the message has been posted.

Nomenclature

WLAN	Wireless Local Area Network
Windows CE computers	Lightweight operative system for handheld
MANET	Mobile Area NETwork
GPS	Global Positioning System
RMPD	Rapid Mutual Peer Discovery
AD HOC Network consisting of	A decentralised and self organised network autonomous nodes.
XML	eXtensible Markup Language
eVC++	embedded Visual C++
UDP	User Datagram Protocol
TCP/IP	Transmission Control Protocol /Internet Protocol
802.11b	IEEE standard for Wireless communication
GSM Codec	Algorithm used for compressing speech in the cellular GSM network.
Mbps	Megabit per second
MBps	MegaByte per second
3G	Third generation cellular phone net

1 Introduction

My master thesis is a part of the Monger project at the Interactive Institute in Stockholm. The aim of this project is to design an application that enables road users to share important information with fellow road users.

The basic idea about Monger is that every driver possesses important local information about the roads they are using. By sharing this information, other drivers will be aware of possible complications in the traffic.

Today road users report to paramedics when an accident has occurred and the traffic information central distributes the information via FM-radio band so that all drivers that listens to the radio are aware of the information . With the Monger application, it will be possible to report directly to other road users in the vicinity of the accident.

A requirement for this kind of application is that you should be able to associate a message to specific locations and also be able to share them to other users in the vicinity.

It is of big interest to design an application as the Monger application without a server or central that distributes the messages. Having a solution in 3g or other similar techniques would be too expensive since it would be almost impossible to keep track of all users current location and trigger the appropriate messages This raises many interesting questions, above all in the design of the application.

My tasks in the Monger project will be to design and implement a suitable distribution algorithm, handling the message transmission between nodes and handling the storage of the messages.

1.1 Background

The starting point of the Monger application is a former project at the mobility studio at the Interactive Institute, called PlaceMemo[8]. The PlaceMemo application offer the users to record a voice message, a PlaceMemo, and associate it to a specific location with GPS data. When the location where the message has been left is passed the information will be played. PlaceMemo is specially designed for an occupational group, road guards, who have great interest in each others placememo, since they often work in a distributed manner and seldom meets along the roads. PlaceMemo distributes messages with a server based solution, which guarantees the distribution to all interested parts.

Monger will be designed for the general road user, but it has many similarities with the PlaceMemo application. The main difference between the two applications will be the way Monger distributes messages without a server solution.

1.2 Problem definition

Road users are often in need to communicate with other road users in their vicinity. Today people hoot, wave or flash their headlights to each other in order to warn fellow road users. Often it is hard to convey a message with this kind of communication.

One solution to this issue is to use Mobile Ad Hoc Networks (MANETs) to distribute voice messages among road users. Distributing information in MANETs is an interesting research area. Most of the research that has been done within this area has handled message distribution to all nodes within a system or routing a message to a specific node within a system. Our task is to design a distribution algorithm that only distributes and presents local information locally, i.e. to people that are interested of the information.

1.3 Outline of the report

This report is split up in well defined sections. Each section is representative for processes and work I have gone through during the project.

In chapter 1 I will start with describing the working progress of the project and give some background information about the project. In chapter 2, the technical equipment and software used in this project will be presented.

Chapter 3 describes the actual Monger application.

In the subsequent chapter 4, the design of the Monger application is discussed. We also present articles that has worked as a source of inspiration to the design of the application.

Chapter 5 presents the technical implementation of the code and how I have solved technical issues.

Chapter 6 concerns the evaluation of the prototype. Chapter 7 presents some problems that has occurred during the project and the final chapter presents conclusions and give some hints about future work that could be done on the prototype.

1.4 Working Progress

During the whole project I have had close co-operation with Mattias Östergren, Ph. Doctor student at the mobility studio. Mattias has acted both as a supervisor and as a member of the project. His experience of similar projects has been of great advantage.

To reach the goal of the project, the project has been divided into five phases. Each phase will be further explained. The different phases are:

1. Pre Studies
2. Designing the application
3. Implementing the code

4. Evaluation of the application
5. Conclusions and Finish the written report

Starting the monger project at the Interactive Institute, demanded extensive pre studies. Several papers were read about different distribution algorithms in MANETs[2]. The pre study phase also included reading and elaborating with C++ and Pocket PC to be familiar with the development tool used later in the project.

The following phase was to take advantage of the information gathered in the pre studies and design the prototype. The main part of this phase was to design a distribution algorithm, suitable for our needs. This phase was done in close co-operation with Mattias Östergren. Several of hours and meetings were spent to bring forth the distribution algorithm. This phase contained elaboration with different types of parameters, i.e. life time of messages etc. that we included/excluded to find out a proper algorithm.

In the fourth and most time consuming phase, the actual coding of the application was done. We divided the implementation of the code between each other. This phase raised several questions, mostly a consequence of my restricted knowledge in programming.

After the implementation it was thought that a small evaluation should take place to see if the prototype works in real life and if there are any failures in the design of the application. Unfortunately we didn't manage to deliver a workable prototype within this project.

The last and most administrative phase was to finish the written report.

[2] Mobile Ad hoc NETworks

2 Technical equipment

This section will present the technical equipment used in the Monger project.
The Monger prototype is the first phase of an iterative process to develop a complete communication system and the reader should have in mind that this first prototype is the first step against a complete solution

2.1 Hardware

We believe, that in the future, a majority of the cars will be equipped with integrated computers which can serve as communication channel to the Monger application.

In this first phase we have chosen to work with handheld computers equipped with integrated wireless network card and externally connected GPS Receiver as platform to the application.

2.1.1 Personal Digital Assistant (PDA)

The central part of the hardware used in the Monger project consists of a handheld computer. Characteristic for a PDA is the small size, pretty large memory and its touch screen. The PDAs, as the name claims are often used as digital calendar and not as computers. Many of the models on the market today have colour screen and built in WLAN(Wireless Local Area Network) cards that makes it possible for users to browse the web or communicate with other users through ad hoc networks.

We have chosen to develop the monger platform on Compaq iPaq 5450[3], which have a built in WLAN card, supports bluetooth communication and like all Pocket PC products offer a free development environment for applications.

The PDA offers us a channel to send and receive messages in the monger application. It also supports the presentation of visual messages on the screen or oral messages as sound messages in the loudspeakers.

2.1.1.1 Pocket PC

Pocket PC is basically the same as Windows CE operating system with some small alternations. Windows CE was created from scratch and not as many people think, a stripped down version of Windows 95. The reason was that Microsoft needed an operating system for small devices that didn't need the boot time on start up.

When switching off a Windows CE unit , it keeps running in a low power state in the background. This means when switching on the unit again you

[3] www.hp.se

are just waking up the unit rather than switching it on from scratch. This is quite an important functionality for small devices such as PDAs.

Pocket PC is designed for devices like handheld computers and electronical organisers. Earlier the operating system supported many processors, but has lately concentrated only on the ARM processor.

2.1.2 GPS Receiver

Compaq iPaq 5450 provides necessary hardware to support playback and recording of voice messages. It also provides limited amount of storage. There are although one crucial functionality that the 5450 lack to support; Positioning. To be able to associate positioning data to a Monger message, we need a GPS receiver. There are many receivers available on the market with different functions and accuracy. We have chosen to connect an external GPS-receiver through bluetooth to the PDA (BTGPS). In a near future the GPS-receiver will hopefully be an integrated part of the PDA or the car.

2.1.2.1 Retrieving the GPS information

The GPS receiver retrieves GPS information to the PDA about the position of the user and makes it possible for the monger application to associate positioning information to a Monger message. The GPS receiver, has to retrieve information from minimum 3 out of 24 satellites that circles around the earth. Information from one satellite locates a sphere on the earth where the position is located. Using two satellites retrieves a circle. The searched position is somewhere on the circle. Using a third satellite specifies two points on the earth, where one can be excluded as improbable. Three satellites are the minimum amount of satellites to retrieve the position. Out of the 24 GPS satellites in the space, 5 are always visible, where ever you are located on earth. The position is retrieved thanks to triangulation and the accuracy is between 20 mm and 15 meters. The inaccuracy depends mainly on reflection of the GPS signal on buildings etc. which causes the signal to take longer time than supposed.

2.2 Software

There are different kinds of programming environment available that supports both Windows CE programming and C++ programming. We have chosen to work with eMbedded Visual C++. eVC++ is an integrated development tool for Pocket PC applications and is developed by Microsoft. eVC++ is a free development kit and is well suited to our demands. It provides resources for both Pocket PC- and C++ -programming. The whole application has been implemented in C++, which is an object oriented programming language. We have also used the platform independent language XML to develop our database.

3 The Monger application

The monger application is a peer-to-peer application for PDAs capable of wireless Mobile Ad hoc Networking (MANETs). The starting point of the whole Monger idea is that road users possess important information about the traffic and the roads that they are using. Road users are the first that reach the scene of an accident or discover obstacles on the roads. Instead of just calling the police and ambulance when an accident occurs, the road users can alert other road users in the vicinity of the scene of the accident. When other the road users are in the proximity to the accident, they will be alerted.

The distribution of the messages will take advantage of peoples every day motion in the traffic and the distribution/exchange of messages will take place during traffic encounters, which are very spontaneous and often occur during very limited time slices. This puts heavy demands on the design of the Monger application.

A monger message is a voice message appended with information about the position where the message was put. The position is gathered with an external GPS receiver connected to the PDA.

We will illustrate the application with an example:

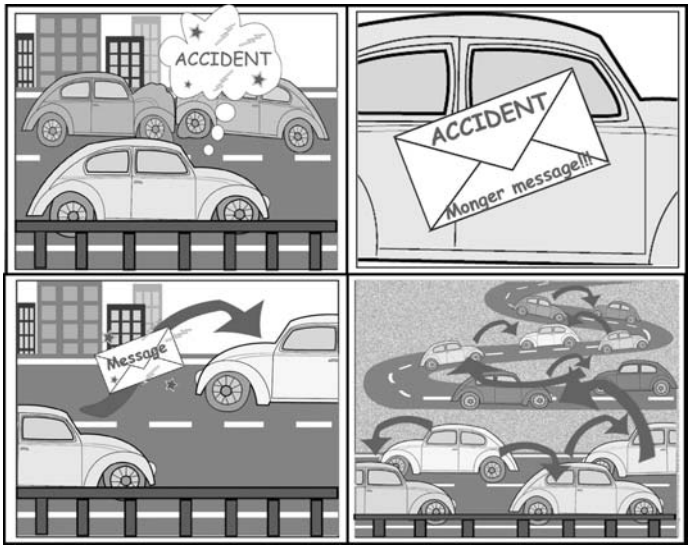

Fig. 1. Illustration of the Monger application.

1. A driver passes a location where an accident has occurred.
2. To warn other road users, he submits a voice message to the Monger application.

3. When he encounters anyone, also equipped with the Monger application, the message and all other messages that the fellow road user don't have in memory are sent over to the oncoming driver.
4. If the oncoming driver passes any location where a message in memory has been put, the message will be played and thereby alert the driver.

The application should be thought of more as a complement to the Traffic Information Systems (TIS) of today rather than a replacement. We argue that there is interest in information systems that distribute local information locally since most of the information distributed in Traffic Information Systems today are of local character.

4 Designing the prototype

In this section we will present design challenges that we have considered during the design phase of the application.

The Monger application has been designed in a modularised way. Each module handles different issues. The reason for designing in a modularised way is not only that it is easier and more convenient to implement, but it is also easier to complement the application with further features in the future.

Below I will describe how we have designed the application and hopefully also give answers on why we have done it. The reader should have in mind that monger is a prototype and not a completed product, which explains why our solutions might not always be optimal solutions, instead they often are the simplest solutions to fulfil the requested task.

4.1 Discovering other users in a MANET

Since the application will be designed for use in traffic it is of great importance to discover when new nodes are within radio range. The faster this process is, the more time does two nodes have to communicate with each other and exchange messages within the restricted time slices that traffic encounters offer.

During an encounter in 70 km/h the nodes are travelling in 40 m/s relative to each other. The radio range for 802.11b is about 300 meters outdoors, which means that the nodes have 600 meters of radio contact if they encounter from opposite directions. This means that the time available for communication is 600/40 = 12.5 seconds. Data is compressed to about 2 kB/s with GSM codec and each message will approximately be 10 seconds long which means that each message will be 20 kB. Theoretically the transmission rate for 802.11b is 11 Mbps, in practice a transmission rate up to 5 Mbps (0.625MBps) is to be expected. This means that during an optimised encounter a maximum of 0.625/0.02 • 31 messages could be interchanged.

To fast discover if new nodes come within radio range we use a protocol called Rapid Mutual Peer Discovery (RMPD). The RMPD protocol has been developed and used with success in former projects at the Institute [1,2]. The creator of the protocol is Mattias Östergren.

The structure of the protocol used in this project is similar to earlier projects at the institute [1,2,3] with minor changes. One small technical change is that we have used STL (Standard Template Library), which is a C++ library of container classes, algorithms, and iterators. Using STL has made it considerably easier to implement the RMPD.

The main issue for the RMPD protocol is to report when new nodes come within radio range or fall out of radio range.

4.2 Algorithm for distributing messages in a MANET

The distribution algorithm must support eventual delivery of messages to the destination without any assumption on the underlying topology and connectivity of the network. In fact only periodic pair-wise connectivity can be required considering the environment where the Monger application will be used. The distribution algorithm should also be designed so that as little "overhead" information as possible is interchanged between two nodes before the actual message exchange takes place.

Designing a distribution algorithm for use in traffic is a crucial task. There are many questions that has to be taken under consideration. Among them is the limited time for messages to be exchanged during traffic encounters. Below I will start with describing some related work that has been investigated to find a suitable distribution algorithm and finally I will present the final distribution algorithm that we have designed, suited to fit our demands.

4.2.1 An Epidemic Model for Information Diffusion in MANETs [4]

Mobile ad hoc networks (MANETs) are constituted by mobile devices equipped with short range radio transmission. The mobility of the nodes raises problems in distributing messages. In traditional network topologies flooding is an approach to distribute messages to every node within the network. Flooding exposes some unnecessary overhead, but it provides a robust strategy for information dissemination. In MANETs there are occasions when the information dissemination doesn't work as expected. These occasions depend on the nodes density within the network. To many nodes leads to broadcast storms and to few nodes will stop the flooding process.

In this paper the authors investigate a dissemination strategy depending on node density. They use a model with similarities to epidemic diseases. When a node discovers other nodes, it advertises a summary of its entities. The listening nodes then request the information entities they are interested in. If a node carries the information, it is infected, otherwise it is susceptible. Conclusion of this paper is that the epidemic algorithm works well when the density of nodes are optimised. Problems also occur when a network is open[4], then you have to introduce some functionality to keep he messages from spreading, so that the information wont spread to the infinity.

This paper is closely related to our problem statement and the distribution algorithm presented seems to fit many of our demands. If we chose to use a similar distribution algorithm we need to introduce a suitable death certificate since our system is open.

[4] Open network: network that consists of a non decided amount of nodes.

4.2.2 GPS-Based Message Broadcast for Adaptive Inter-vehicle Communications[5]

In this report two new broadcast protocols that make use of GPS information to enhance the performance of broadcast services in IVC (Inter Vehicle Communication) are proposed. Traditional broadcast protocols create too many redundant messages and it doesn't take the position of the vehicles in consideration. The first protocol proposed, the TRADE (TRAck DEtection) protocol, categorises the neighbouring vehicles into different road groups (Same_road_ahead, Same_road_behind, etc.) and chooses a few vehicles in each group for message re-transmission. This shows out to be a poor solution since every node has to know the GPS information of all other nodes, which consumes a lot of bandwidth.

DDT (Distance Defer Transmission) protocol, is the second broadcasting protocol proposed. DDT includes its current GPS position in the broadcast message. The basic idea of the DDT protocol is timing. When a message is received, the defer time for retransmission is calculated inversely proportional to the distance from the sender. A receiver further away retransmits before a node closer to the sender. If a node is far away, it is selected as a border vehicle and immediately re-transmits the message. Otherwise it determines if retransmission is necessary after gathering position information about neighbours that transmit the same message.

This paper presented some interesting ideas about how to use the GPS data to optimise broadcast messages. This is not of interest in the Monger project, but it gave us some ideas about how to integrate the GPS in the distribution.

4.2.3 Pollen: using people as a communication medium [6]

Pollen networking provides an infrastructure that supports distribution of information without relying on traditional network technology. The idea behind Pollen network is to use peoples every day action instead of wires to network devices, artefacts and physical places. When a user passes a place where information (pollen) is sent to the users device, the pollen may be either visible or invisible to the user, depending on the relevance to the user. Pollens that are not relevant to the user are passed to other users when they come within radio range. Complement to the people-based network is a central organisational memory. When a user's PDA is docked with his workstation, the cached pollen is uploaded to the central organisational memory. Information can also traverse the opposite way i.e. from the central memory to the people people-based network. Information in the network can be passed between two arbitrary nodes in several ways. It can either directly pass from one node to another. It can pass through intermediate nodes or it can be passed via the centralised memory.

Pollen networking is not suitable for applications that demands fast response or guaranteed delivery.

In this paper the authors used the algorithm rumour spreading to control the exchange of pollen. The simulation showed that you could expect 63% of the nodes in a sufficient large network to be pollinated in log(N) time, where N is the number of nodes.

Pollen networking uses people as communication medium. This is very similar to our idea, to use road users as communication medium. One big difference between Pollen and Monger is that we are not interested in sending messages to a specific host within the system. We are only interested in distributing messages to nodes in the vicinity.

4.2.4 Epidemic Routing for Partially-connected Ad Hoc Networks [7]

Existing ad hoc networks assume a connected path between source and destination. This paper handles cases where there never is a connected path between the source and destination e.g. disaster areas, military deployment etc. To handle this problem they investigate epidemic routing as a possible solution.

The goals of epidemic routing are:

i) maximise message delivery rate
ii) minimise message latency
iii) minimise the resources consumed in message delivery.

In partitioned networks, epidemic routing relies on nodes coming in contact with other network partitions through node mobility. To accomplish the goals of epidemic routing stated in this paper, they use an upper bound of message hop count and limitations on per-node buffer space (the amount of memory devoted to carrying other host's messages.) For efficiency each node has a list of messages that it is buffering as well as messages that it has originated. A hash table indexes this list. Each host has a bit vector called summary vector that indicates which entries in their local hash table are set. When two nodes come in contact with each other they compare their bit vectors and the one with the smallest bit vector initiates an anti-entropy.

To reduce redundant connections each node maintains a cache of hosts that it has recently spoken with.

In the simulation of this work each simulated mobile node has an epidemic routing agent layered on top of the IMEP (Internet MANET Encapsulation Protocol) protocol.

The IMEP layer is responsible for notifying when new nodes come within/ move out of radio range. In the worst case when the radio range were 10 m, the result was that 89.9% of the messages were delivered. In order to guarantee eventual message delivery, a subset of nodes has to have buffer space equal to the maximum number of messages at any given time. It is also possible to achieve robust delivery rates with less buffer space if the nodes have different buffer capacities. The final result of this paper is that epidemic routing delivers 100 % of the messages with reasonable resource consumption where existing ad hoc protocols fail to deliver any messages at all because no end-to-end routes are available.

This paper also include investigation of epidemic routing and it adds a restricted amount of memory as a parameter when evaluating the algorithm.

An environment with no guarantees on connections between source and destination is very similar to the monger case. One difference although is that we do not demand that messages are delivered with 100 percent certainty. The idea of exchanging a list of which messages the node have in memory is very interesting and should be further investigated.

4.3 Algorithm designed

The algorithm we have chosen for distribution of messages in the system basically uses a protocol called epidemic routing or rumour mongering. This distribution model has been investigated in several papers, and it seems to be well suited for MANETs, where the nodes often move and connections between nodes are very spontaneous. Related work that has investigated dissemination in MANETs (Mobile Ad hoc NETworks), have had a slightly different approach to the problem. They have been interested in routing messages from one source to one destination. In the monger project we are not interested in sending messages to specific users, instead we are interested in distributing local messages locally in the system.

Epidemic routing, as the name claims, has similarities to epidemic diseases. When a node discovers other nodes, it advertises a summary of its entities. The listening nodes then request the information entities they are interested in. Problems occur when a network is open, as the Monger network, then you have to introduce death certificates so that the messages won't spread to the infinity [4]. With death certificates we mean a way to tell the rest of the nodes in a system that a message is deleted (dead) and should not be forwarded any more. A death certificate can either be a timestamp or a death message that is distributed to all nodes in the system.

Our distribution algorithm is very similar to the distribution algorithm proposed in "Epidemic Routing for Partially-connected Ad Hoc Networks" [7]. We have made minor changes to the algorithm, but the conceptual idea is the same. We will exchange a list of messages that are in memory, before the actual exchange takes place and also have a module that alerts when new nodes are in the vicinity. The best way to explain our solution is to illustrate the algorithm with an example:

Node A[5] is driving along a road. The roads are treacherously icy and Node A leaves a message to warn other road users that will pass this certain spot. When Node A discovers an other Node B within radio range, Node A sends an indexList of all messages that A has in memory to node B. B compares the indexList received with its local indexList, i.e. Node B:s indexList. Node B creates a wishList by picking out those indexes from the received indexList that aren't in the local indexList. The wishList is sent back to Node

[5] A user of the Monger application.

A. When Node A receives a wishList, it traverses the wishList and sends over one message at a time to node B. Node B saves the messages and updates its indexList.

4.3.1 Lifetime of messages

Considering that the Monger application is an application built to report and distribute local messages that are only of interest for a restricted time on a restricted area, one of the major issues is to decide how long the life time of a message should be. Another issue is to decide whether to use hop counter or time to decide when to eliminate the messages. Using a lifetime that is to short will cause many messages to be discarded before they have been distributed in the system. On the contrary, using a lifetime that is to long will not only cause too many messages within the system, but also messages that no longer are of interest will be distributed. To decide which lifetime we should chose we have to decide for what kind of circumstances we will design the application. With circumstances, we mean what milieu e.g. urban area or in the country etc.

Considering the type of the messages, location dependent and time restricted messages, that the monger application intend to distribute, we have chosen to design the application for urban areas. Take for example the situation when a deer runs out on the highway in the country side. You want to post a message to other users in the system to warn about the deer on the road, but you wont meet anyone in the following five minutes. Is it then an idea to warn about the animal? With this example we wanted to exemplify the problem with using the Monger application in the country side. To find out if the application is possible or well suited for use in the country side, we should simulate the traffic and se how messages are distributed. Unfortunately the limited time of this project does not allow such simulations. In the urban case, we assume that the traffic encounters are sufficiently frequent that the Monger application will perform well. We have chosen to design the application for the urban case.

It should be mentioned that it is easy to change the lifetime of messages in the application, so we are not strictly bound to urban areas.

Since simulations are outside the scope of this project, a lifetime of one hour will be used. The reason for choosing one hour is that we think that it is sufficient large amount of time to distribute a message in the system, and it wont be out of date within one hour. Of course there will be situations where this time is too large or to small, but it would be impossible to suit the lifetime to every single occasion. One possibility we have considered is to have the distance from the posting point as a parameter inverse proportional to the willingness to spread messages, which could be an option for future work on the prototype.

In that case, messages that are positioned far from the current location, will have lower priority to be distributed than messages that are near the current location and messages of interest will be prioritised.

An other solution would be to use TTL[6]. For the Monger application it is very hard to find an appropriate maximum hop size since the circumstances under which the Monger application is used can vary heavily.

Two examples where it is hard to chose an appropriate TTL are:

- A user is in a line of cars and there will be many hops within a short period of time, which means that the message will die out early.
- A user is encountering fellow road-users very seldom, which means that old messages will be distributed in the system and it will take long time before messages dies.

4.3.2 Database design

PDAs are practically stripped down computers with only basic functionality compared to stationary computers. This of course makes impact on the performance of the PDA. The PDAs used in the Monger project are Compaq iPaq 5450, with a 64 MB RAM memory. The limited amount of memory puts demand on the storage and format of the messages.

There are different ways to implement Databases in Pocket PC/Windows CE. Windows CE has for instance a built in support for a set of database APIs, but there are also other solutions available. I will describe some of these in the following section

The built in database is simple, with only one level and a maximum of four sort indexes. It is a good choice for uncomplicated data such as address lists etc. A Windows CE database is composed of a series of records. The records can contain one out of nine different types. A restriction is that records can't contain records and the database can't be locked, but there are ways to notify a process that another process has modified the database.

Each sort index results in a fair amount of overhead in the database and it is preferable to hold down the amount of sort indexes.

An other way would be to use a database in SQL. SQL stands for Structured Query Language and is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" can be used to accomplish almost everything that one needs to do with a database. SQL is available for Pocket PC devices, with minor restrictions compared to ordinary SQL.

A third way is to use XML. XML stands for eXtensible Markup Language and is originally developed by the World Wide Web consortium (w3c). The

[6] TTL = Time To Live

strength of XML is that it is a common standard to organise information in such a way that it can be interpreted and processed without dependency to application or operation system.

XML has many similarities with other markup languages like HTML, XHTML etc. XML, however, is a meta-markup language. It's a language in which you make up the tags you need as you go along. These tags must be organised according to certain general principles, but they're quite flexible in their meaning. XML tags describes the information in a document, not the look of the document. Since XML was first released in 1998, it has gained much popularity, partly because of the simplicity of the language, but also because the ability to structure information in a simple and clear manner. XML is today used in a wide range of different applications.

On the basis that the built in database has been used in former projects at the Institute[8] and evaluations has shown on complications with unreliable performance, we have chosen not to use it in the Monger application.

The choice fell on XML, mostly because we didn't find any appropriate SQL solution for Windows CE and also because XML:s platform independence characteristic.

To find information in a XML document we have to use a parser that traverses the tags in the document and finds the information requested. There are several different parsers available on the market. Our choice fell on MSXML. MSXML is a free API developed by Microsoft, which is used to parse XML documents. We have used the memory mapped hierarchical tree-based API, DOM, within MSXML, witch means that we load the whole document into memory when we work with the database. This might not be an optimal solution, because of the restricted amount of memory in the PDA. We considered to work with SAX API instead, which only loads the information of interest into memory. SAX is much more memory efficient, but the effort of programming SAX exceeds DOM. Since we are working with a prototype, we chose to work with the easier interface, DOM. In later versions of the prototype SAX should be considered as a solution, but we then have to study the possibilities to change data in the database, which today is a bit awkward.

The Monger messages are stored in the Database as wave files. The database also stores the length of the message, the position and time where the message was left and a reference of where in memory the message actually is.

The structure of the XML document is as shown in figure. 3

<DataBase>

<Monger_Msg lat=1, long=2, time=3, ref="//monger/msg1", Length=10\>

<Monger_Msg lat=5, long=7, time=8, ref="//monger/msg2", Length=20\>

</DataBase>

Fig. 3: The structure of the XML document used in the Monger project

Searching the database every time we get a new position or change speed etc will consume a great deal of the system resources. We have therefore used an algorithm used in PlaceMemo, with only minor changes. To avoid using a great deal of system resources, we use a safe perimeter, that calculates the distance to find the nearest message to our current position. The distance calculated is the longest distance we can move without triggering a message.

As long as we do not proceed outside the safe perimeter no additional database searches will be necessary. Each time we check the database for any hits, we start by calculating if we have exceeded the safety perimeter. If we have, we check if we have hit any messages This is a decided improvement and reduces the database searches significantly.

To further improve the database search algorithm it is important to design the database as efficient as possible.

We have chosen to create the database with the latitude as search order. Below we will illustrate how the database works.

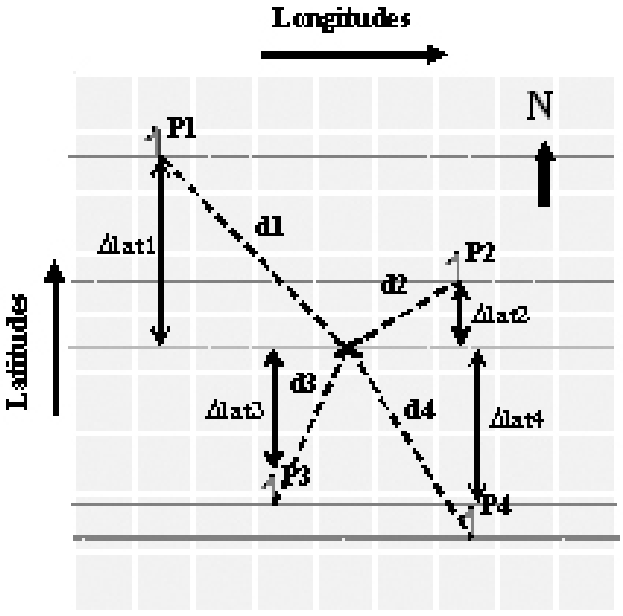

Fig 4: The database search algorithm

As the database search begins, we set our seek pointer onto the record with the largest latitude value smaller than our current latitude. Our current position is marked by the X in the figure. The seek pointer will in this example after the operation point to record P3 in figure. The algorithm continues to search the records with the closest latitude values on either side of record P3 calculating the distance to each coordinate. Once the distance latitude-wise exceeds the shortest distance found so far the search is completed.

In the above example P2 will be the closest position. Even if we imagine there to be many more records in the database containing latitude values larger or smaller than the ones in the example we are able to conclude that P2 is the closest and thereby end the search. This conclusion can be made since the latitude difference between the current position and P1 or P4, lat1 and lat4 alone exceeds the distance to P2 (d2).

The distance to the closest memo is, after the search is finished, used to set the safe perimeter or decide if the message is to be trigged as described in previous sections. It is also possible for many messages to be on the same latitude, but on different longitudes. The calculation of the safe perimeter takes this into calculation by always checking if there are messages on the same latitude and if the distance is shorter than the distance to the nearest message on an other latitude.

4.3.3 User interaction

The Monger application will be designed for use in traffic. It is therefore of great importance that the interaction demanded when submitting and receiving Monger messages is very user friendly. The drivers attention shall always be on the road and not at the PDA.

The Monger application, will as mentioned earlier be designed for use in traffic. It is therefore important to design the application with a very user friendly interface. Because of the restricted amount of memory, written messages would be preferable, but it takes to much of the drivers attention to post a written message to the system. It has to be easy to record and receive messages. It is troublesome enough to write information on a PDA while not driving. It would be a danger to traffic to have to write in messages while driving. The most suitable input method is voice messages. They indeed take much more place in memory, but we think that this is the only possible way to present messages in such an environment. Since storing audio requires much memory, voice messages have to be compressed before they can be saved. The compression coder/decoder used, is the platform-included Microsoft's GSM codec [9]. The GSM codec is based on the same algorithm that is used for compressing speech in the cellular GSM network. The algorithm is very specialized to compress speech so it will not work well when used to compress sounds other than speech. Any noise lacking the characteristics of speech will therefore affect the performance negatively. Data is compressed to about 2 kB/s with GSM codec.

To record messages, the hardware button on the device must be pushed. The recording may be stopped by either pushing the hardware button or it will automatically be stopped after reaching an upper limit of time. The time restriction of message length is to avoid too big messages in the system. It would be hard to distribute very big messages during traffic encounters, which would lead to a poor application.

Messages are automatically played when a position where a message has been posted is passed (if the message is in memory), which means that play back will not demand any interaction with the user.

To strengthen the presentation of a message, an image with a yellow warning triangle will pop up on the screen each time a message is presented. This can be useful if the PDA has no sound on or to pay visual attention to the user that a message is presented.

5 Technical implementation

This section will present how we technically have designed the Monger prototype. The application is split up in several modules, each module handles different tasks. We have chosen to implement the code in a modularised way, which makes the program scalable and easy to implement.

The modules are:

- MongerLib
- Scheduler
- RMPD
- Positioning
- IndexList
- DataStorage
- Communication
- Audio

5.1 The MongerLib module

The MongerLib module is a Dynamic Link Library (DLL), which is the starting point of the application.

A dynamic link library takes the idea of an ordinary library (also called a statically linked library) one step further. The idea with a static library is for a set of functions to be collected together so that a number of different programs could use them. This means that the programmers only have to write code to do a particular task once, they can use the same function in lots of other programs that do similar things. However, with static linking the linker builds a program from all the object files that contain functions or data used in the program, which includes any library functions that are used. Each program gets it's own copy of all the library functions built into it.

A natural extension of the idea of a library is to put all the code for the functions in a library, but do the linking when the program is run instead of at link time (thus it is "dynamic"). A dynamic link library is a lot like a program, but instead of being run by the user to do one thing it has a lot of functions "exported" so that other programs can call them. There are several advantages to this. First, since there is only (in theory) one copy of the DLL on any computer used by all the applications that need that library code, each application can be smaller and save disk space. Also, if there is a bug in the DLL a new DLL can be created and the bug will be fixed in all the programs that use the DLL just by replacing the old DLL file. The MongerLib DLL starts a thread where the Monger application is running.

5.2 Scheduler

To execute tasks at specified times, we have used a scheduler that schedules tasks in the process. You can think of the scheduler as the operative system of the application.

The scheduler also consists of listeners. A process can use these listeners to be aware of when information they are listening to changes. This reduces a lot of process time. Instead of having each process checking the state of some specific data of interest, the scheduler notifies all interested listeners when data has changed.

5.3 Rapid Mutual Peer Discovery[7]

In order to accomplish rapid mutual discovery in a group of peers we have used an algorithm originally developed for the hocman project[2].

Each node keeps a list of known peers. Each entry in this list contains connectivity information, such as which port that peer's Monger application listens to and its network identifier. Each entry is also associated with a timer. The entry is removed and the corresponding peer is deemed unreachable, when it times out. At a regular interval, but slightly jittered to avoid synchronisation, each peer announces its presence by broadcasting a hello message, containing the connectivity data for that peer. When receiving a hello message from a peer present in the list, the timer of the corresponding entry is reset. Otherwise, an entry is created and appended to the list of known peers.

It takes x seconds before mutual discovery is established with a hello-message broadcast rate set to x seconds. The solution to speed up the process is to broadcast a reply message whenever a new entry is inserted in the list. This reply message contains connectivity information of the peer that is about to send it (A) as well as connectivity information of the peer that was inserted in the list (B). When other peers receive this reply message, they check the following condition: is A unknown and is either B present in the list, or is B this peer? If so, then the receiving node can safely assume it can reach peer A and appends it to its list. However, it refrains from sending a reply message. If the condition does not hold, the reply message is ignored.

The following example illustrates the message exchange in the algorithm. Consider the case when three peers A, B, C come within wireless reach simultaneously. Assume Peer A broadcast a hello message announcing its presence. When Peer B and C receive the hello message from A they update their respective list. Assume Peer B send a reply message before C, indicating it has received the hello message from A. Peer A and C receive the reply and update their respective list. Finally, Peer C sends a reply message indicating it has received the hello message from A. Peer A and B updates their respective list on reception. No more messages are sent since all nodes are aware each other.

[7] http://www.tii.se/mobility/Files/camera-ready1.pdf

5.4 Positioning

This module handles and presents the GPS-information received. It is possible to receive the information in cartesian coordinates or in radians. This module also verifies if a message shall be played, e.g. if our position is identical to any position in the memory.

5.5 IndexList

Traffic encounters offer a restricted time to exchange information. It is therefore important that we optimize the exchange phase and transfer data that is of interest to both parts in an encounter. We would get a poor system if we tried to transfer all messages in memory each time we encountered another car. This would distribute messages located first in memory and give lower priority to messages further back. An other solution would be to pick out a message randomly in memory and transfer it, which gives all messages the same probability to be exchanged, but this also has its drawbacks.

Our solution is to have an IndexList consisting of entries of all messages in the memory. The IndexList is exchanged and each node chooses which messages they are interested in and creates a WishList. This causes some overhead information exchanged, but on the other hand, only information that is of interest is exchanged.

5.6 Communication

UDP Sockets are used to communicate between nodes in the Monger application. UDP communication is an unreliable communication method. Data that is lost is not resent, like the case in TCP/IP communication. The reason for choosing UDP is that we don't have time to set up a TCP/IP communication session each time we want to exchange data. This would take to much processing time. Further, the gain in time saving in not resending lost information is greater than the loss of quality in the voice message. Often, it is possible to interpret a voice message even though not all the information is received.

6 Problems in designing the application

The design process of the Monger prototype has been very interesting. Starting from scratch and follow all steps from designing a prototype on paper to implementing and evaluating the prototype in the end has been very instructive.

During the project, many complications has arisen. The greatest complication has been to implement the code. Since both Pocket PC and C++ was new programming languages for me, this process was very drawn-out and the technical solutions might not be the most optimal solution.

Unfortunately, this drawn-out phase of the project has made it impossible to develop a working prototype because lack of PDAs that were needed in other projects at the mobility studio.

One major technical problem was that Microsoft had a bug in their MSXML load function[8]. This bug caused a major delay in the implementation phase

There has also occurred organisational problems during the project which resulted in restricted help from my supervisor at the Interactive Institute.

[8] http://support.microsoft.com/default.aspx?scid=kb;EN-US;301935

7 Conclusions and future work

The intention with this master thesis has been to design a Traffic Information Systems that uses peoples everyday motion in traffic as a communication medium. In this thesis we argue that road users themselves are willing to distribute local information about the roads to other road users in their vicinity.

We have designed a distribution algorithm suited for use in traffic. The distribution algorithm is a mix between epidemic routing and pollen networking. Unfortunately the project failed to deliver a working monger prototype. In a future version of the monger project it would be desirable to evaluate the prototype.
A future evaluation would give answers to two main questions critical for the project.

- Does the prototype work?
- Is the conceptual idea behind Monger workable in real life?

It is interesting to evaluate the prototype in an environment similar to the environment where the application is to be used. Since the application is designed for use in traffic it was obvious to test the application in real traffic situations. The most critical situation for the application performance is when two cars encounters in opposite directions. In this case, the relative speed between the two cars gets bigger which results in a smaller time ,t, available to exchange information, than if the two cars were travelling the same direction.

We should test the prototype in three different situations.

1. Encounters
2. Overtaking situations
3. One car located on a fixed spot, while the other car passes this spot

These situations are representative for most of the situations that can occur. The tests were made in four different speeds (km/h) 30, 50, 70 and 90.

The test will show if the prototype managed to exchange information in all speeds and in all three cases. This small evaluation would verifiy the technical functionality of the project.

To evaluate the conceptual idea behind the Monger application, we would studie how the test pilots apprehended the application.

A future evaluation like the one presented above would indicate if our distribution algorithm is workable for use in traffic. In theory it is. To verify the dissemination of messages within the system it would be very interesting to simulate the spreading in a traffic simulator like the meso

model[9] over Stockholm, which is a micro model of the traffic flow in Stockholm. Unfortunately, this simulation process has been outside the scope of this project. A simulation could prove if the spreading is complete within the system. Other important questions that could be verified are the life time of messages within the system and the maximal amount of data that should be allowed in the system.

In the monger project, we have chosen an arbitrary lifetime that we think will suit the situation. A simulation can prove if the life time is to long, which floods the system or to short, which end the distribution of a message to fast.

A possible evolution of the monger prototype would be to integrate automatic data collection from the car and distribute these messages. An example is the mix of sudden braking and degrees below zero which automatically sends a warning message to the cars in the vicinity. This kind of system has been developed by BMW, but it would be interesting to have both oral messages and automatic messages.

[9] www.movea.se

8 Appendix

8.1 Reference

[1] Östergren, Mattias *Sound Pryer Field Trials: Learning About Adding Value to Driving*. Presented at the workshop Designing for ubicomp in the wild: Methods for exploring the design of mobile and ubiquitous services, at MUM'2003.

[2] Esbjörnsson, Mattias, Juhlin, Oskar & Östergren, Mattias
The Hocman Prototype – Fast Motorbikers and Ad Hoc Networking.
In Proceedings of MUM 2002

[3] Brunnberg, Liselott & Juhlin, Oskar
Velocity and Spatiality in a Gaming Situation – Boosting Mobile Computer Games with the Highway Experience.

[4] Khelil, Abdelmajid, Becker, Christian, Tian, Jing, and Rothermel, Kurt
An Epidemic Model for Information Diffusion in MANETs

[5] Min-Te Sun, Wu-Chi Feng, Ten-Hwang Lai, Kentaro Yamada and Hiromi Okada *GPS-Based Message Broadcast for Adaptive Inter-vehicle Communications*

[6] Glance, Natalie, Snowdon, Dave, Meunier, Jean-Luc
Pollen: using people as a communication medium

[7] Vahdat, Amin & Becker David
Epidemic Routing for Partially-Connected Ad Hoc Networks

[8] Gustafsson, Anton *PlaceMemo: A Prototype of a Context-aware Information System*
Dept. of Computer Science, Umeå University

[9] Nigel Thompson. *Using CODECs to Compress Wave Audio*. Windows Multimedia Technical Articles, MSDN April 4, 1997

[10] http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dnarmulmed/html/msdn_codec.asp , visited 2004-04-28

[11]
http://www.hp.se/produkter/type.asp?category_id=5&type_id=13&audience= , visited 2004-04-28

[12] www.tii.se/mobility/, visited 2004-04-28

[13] http://www.cs.umu.se/~thomash/reports/GPS_files/frame.htm
, visited 2004-04-28

[14] http://www.utsidan.se/utrustning/kartor/gpsintro.htm , visited 2004-04-28

[15] http://epubl.luth.se/1402-1617/2002/141/LTU-EX-02141-SE.pdf
, visited 2004-04-28

[16] www.codeproject.com , visited 2004-04-28

[17] www.codeguru.com , visited 2004-04-28

[18] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk30/htm/xmmscxmldommethods.asp , visited 2004-04-28

[19] www.gamedev.net , visited 2004-04-28

[20] http://www.csd.uu.se/courses/course-material/xjobb/docs-reports/Lars_Goransson-2003.pdf, , visited 2004-04-28

[21] http://www.gamedev.net/reference/articles/article711.asp , visited 2004-04-28

[22]http://www.cegadgets.com/wincedevfaq.htm#1.3%20I%20want%20to%20learn%20about%20Windows%20CE%20development.%20Where%20should%20I%20start**?** , visited 2004-04-28

[23] http://support.microsoft.com/default.aspx?scid=kb;EN-US;301935 , visited 2004-04-28