Dynamic Personal Networks for Location-Based Applications within MediaSense

Joackim Nilsson
Abstract

As the development of context aware applications has evolved, there has been a corresponding increase in need for more sophisticated system. The aim for this thesis is the development of a dynamical P2P network system which is based on locations. The P2P network is self organizing and in a lightweight format. Modern technical solutions including AGPS have facilitated the work associated with the ability to position users and modern mathematical solutions such as spherical trigonometry provides the P2P system with the necessary accuracy even for short distances. The P2P system works on different Java platforms including JSE, JME and Android. Unfortunately the 3G network distributor has not yet solved a NAT traversal problem, which means that the P2P network self organization and architecture has only been proved by means of simulations. Another problem is that certain mathematical formulas are required for the spherical trigonometry and the limitation for JME is that it is unable to handle inverse trigonometry. However, the Android and JSE versions can form a correct P2P network, under the condition that the Android device uses a WIFI connection point outside the 3G distributor network system. This thesis reports the successful testing of the locations-based P2P network.

Keywords: P2P, Context Aware Networks, Context Aware Applications
Acknowledgements

Firstly I would like to thank my tutor Stefan Forsström for his patient guidance and his capacity to answer my numerous questions.

I would also like to thank the MediaSense project at Mid Sweden University for making it possible for me to use the necessary testing equipment for this thesis.
# Table of contents

Abstract ............................................................................................................. ii  
Acknowledgements ....................................................................................... iii  
Terminology ...................................................................................................... 1  
1 Introduction ............................................................................................ 2  
1.1 Background and problem motivation .............................................. 2  
1.2 High-level problem statement .......................................................... 2  
1.3 Scope ..................................................................................................... 2  
1.4 Low-level problem statement ............................................................ 2  
1.5 Outline .................................................................................................. 3  
2 Theory ...................................................................................................... 4  
2.1 Java Virtual Machine .......................................................................... 4  
2.1.1 Java Platform Standard Edition .................................................. 4  
2.1.2 Java Platform Micro Edition ....................................................... 4  
2.1.3 Java Platform Enterprise Edition Five ........................................ 4  
2.2 Android ................................................................................................ 5  
2.2.1 Android Software Development Kit ............................................. 5  
2.3 Positioning ........................................................................................... 5  
2.3.1 GPS positioning ............................................................................ 6  
2.3.2 Positioning using Cell-ID ............................................................. 6  
2.3.3 Mobile Triangulation ..................................................................... 7  
2.3.4 Assisted Global Positioning .......................................................... 7  
2.3.5 Mobile positioning ......................................................................... 7  
2.4 3G ........................................................................................................... 8  
2.5 P2P networking ................................................................................... 8  
2.6 Context aware applications ............................................................... 8  
3 Methodology ........................................................................................... 9  
4 Design .................................................................................................... 11  
4.1 The architecture of the P2P system ............................................... 11  
4.2 P2P-net architecture .......................................................................... 12  
4.3 Geographical Coordinates ................................................................ 16  
4.4 Central Server .................................................................................... 18  
4.5 The P2P system for JME ................................................................. 19  
4.6 The P2P-system for Android ......................................................... 21  
4.7 The P2P-system for JSE ................................................................. 22
# Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.8</td>
<td>P2P simulation system</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>Results</td>
<td>24</td>
</tr>
<tr>
<td>5.1</td>
<td>Application for JME</td>
<td>24</td>
</tr>
<tr>
<td>5.2</td>
<td>Application for Android</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>Application for JSE</td>
<td>26</td>
</tr>
<tr>
<td>5.4</td>
<td>Telia 3G</td>
<td>26</td>
</tr>
<tr>
<td>5.5</td>
<td>P2P net simulation</td>
<td>27</td>
</tr>
<tr>
<td>5.6</td>
<td>Evaluation of Location Based Math</td>
<td>30</td>
</tr>
<tr>
<td>5.7</td>
<td>Location Based Network Evaluation of Connection Roundtrips</td>
<td>32</td>
</tr>
<tr>
<td>5.8</td>
<td>Location Criteria</td>
<td>33</td>
</tr>
<tr>
<td>5.9</td>
<td>Java.Math and JME case 1</td>
<td>33</td>
</tr>
<tr>
<td>5.10</td>
<td>Java.Math and JME case 2</td>
<td>34</td>
</tr>
<tr>
<td>5.11</td>
<td>Memory weakness in Sony Ericsson C702</td>
<td>36</td>
</tr>
<tr>
<td>5.12</td>
<td>P2P System Field Evaluation</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions</td>
<td>38</td>
</tr>
<tr>
<td>6.1</td>
<td>Low level problem formulations</td>
<td>38</td>
</tr>
<tr>
<td>6.2</td>
<td>High level problem formulations</td>
<td>39</td>
</tr>
<tr>
<td>6.3</td>
<td>Discovered problems</td>
<td>39</td>
</tr>
<tr>
<td>6.4</td>
<td>Future Works</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>Appendix A: Documentation of own developed program code</td>
<td>44</td>
</tr>
</tbody>
</table>
Terminology

Acronyms and abbreviations

3G Third generation mobile network
AGPS Assisted Global Positioning System
Android A linux-based mobile device system for Java applications.
DOM Document Object Model
GPS Global Positioning System
I/O Input/Output
JEE5 Java Enterprise Edition 5
JME Java platform Micro Edition
JSE Java platform Standard Edition
JSP Java Server Pages
JVM Java Virtual Machine
NAT Network Address Translation
P2P Peer To Peer network.
Servant A node in a P2P system that is both a client and a server.
TCP Transport Control Protocol
TTL Time To Live
WIFI Acronym for the WLAN standard IEEE 802.11
WLAN Wireless Local area Network
XML Extended Markup Language
XSL Extensive Sheet Language
Introduction

MediaSense [1] is a large ongoing EU-funded research project at Mid Sweden University. MediaSense started in 2008 and concerns “the intelligent delivery of information to any host, anywhere based on context-aware information regarding personal preferences, presence information, and sensor values” [1].

1.1 Background and problem motivation

The use of the Internet and mobile networks has grown rapidly. To meet the increased demand for new services, specialized applications such as context-aware applications and systems have been evolved. Within this field, a new requirement has appeared involving the use of mobile devices with positioning systems. A small local P2P [2] network formation, based on position, can be used in many different applications.

1.2 High-level problem statement

The aim of this thesis work is to use mobile devices in order to form P2P networks based on the location of the user. Therefore the problem for this thesis is defined as:

Is it possible to find a technical solution in the form of a middleware that enables location-based, lightweight and self-organizing P2P networking between different kinds of servants?

1.3 Scope

Because the devices are built on Java platforms, the P2P system is limited to such platforms. The P2P system is also limited to slower TCP low end connections and not to the faster UDP connections. Security and authorization issues both fall outside the scope of this thesis.

1.4 Low-level problem statement

From the problem statement, the goals required to be met and answered by this thesis have been determined:

1. How is it possible to accomplish the middleware to be self-organized, stable and have the possibility to fall back on a central approach, particularly if there are no servants to interact with?
2. How is it possible to determine position of the servants in order to discover whether there other servants fall within the vicinity?

3. Is it possible to set lower end connection of this middleware to a TCP socket type?

4. How is it possible to design the P2P network for different JVM [3] platforms?

5. How is it possible to implement the necessary Java code?

1.5 Outline

Chapter 2 deals with the background theory that this thesis is based on. Chapter 3 describes the methodology used. Chapter 4 is the largest chapter dealing with both the design and implementation of the necessary parts. Chapter 5 presents the results that have been achieved. Chapter 6 is a discussion concerning the results and other aspects relevant to this thesis.
2 Theory

The theory part of this thesis is a presentation of related work in order to provide a better understanding for this thesis.

2.1 Java Virtual Machine

The Java Virtual Machine (JVM) [3] is the runtime system that is the basis for the technological solution for this thesis. JSE, Android and JME platforms have been used for this thesis. The Java language syntax has borrowed heavily from such low level languages as C or C++, but the memory management is handled by means of an integrated automatic garbage collection by the JVM. The Java language also uses references instead of pointers.

2.1.1 Java Platform Standard Edition

Java Platform Standard Edition (JSE) [4] is the standard Java edition, which is also sometimes known as J2SE. In order to run JSE properly a set of libraries or packages are required. JSE is designed for portable applications that run on JVM. The general purpose for the use of JSE is in desktop PCs, servers and similar devices.

2.1.2 Java Platform Micro Edition

Java Platform Micro Edition (JME) [5] contains a subset of the packages used by JSE. Applications are called midlets because of the use of the package javax.microedition.-midlet, which defines the Mobile Information Device Profile applications (MIDP). The reason for using only a subset is because the mobile devices that use JME have very limited memory storage capacities.

2.1.3 Java Platform Enterprise Edition Five

Java Platform Enterprise Edition Five (JEE5) [6] is used for the browser part, this edition has all the packages available to JSE but JEE5 also handles server scripting such as servlets and JSP pages.
2.2 Android

Android [7] has been developed based on an initiative of Google for the Open Handset Alliance and it is an open platform for mobile devices, which uses Android for its operating system, middleware and some key applications. Android is based on the Linux Kernel.

2.2.1 Android Software Development Kit

Android Software Development Kit (Android SDK) provides the necessary tools in order to develop applications on the Android platform and has an API [8] with many of the features in JSE. However, every application runs its own virtual machine called Dalvik [7]. The Dalvik virtual machine is register-based as opposed to the JVM which is stack-based and it executes files in a special format called Dalvik Executable (.dex), which is optimized for mobile devices with a small memory capacity. Many of the packages available in JSE are also available in the Android platform.

2.3 Positioning

Since the shape of the earth is not a flat surface, formulas such as Pythagoras Theorem for flat areas are not applicable if accuracy is required within the calculations. What is required, are particular laws of trigonometry for spherical objects [9]. Even if spherical trigonometry is considered, the earth itself is not a perfect sphere. It is well known that the earth has its longest spherical radius at the equator and its shortest at the poles. The radius is 6378.1 km for the equator and for the poles it has a radius that is 6356.8 km [10]. The use of latitude and longitude measures the angles that are relative to particular point on the earth’s surface as shown in Figure 2.1. The latitude measures the angle between the radius that points to a given spot on the surface and the corresponding radius on the equator. The longitude measures the angle between the radius pointing to a particular that spot on the surface and the corresponding radius on the Greenwich meridian.
Dynamic Personal Networks for
Location-Based Applications -
within MediaSense
Joackim Nilsson

2 Theory
2010-05-19

In order to perform spherical trigonometry calculations, formulas such as the Haversine formula [9] have been used. This formula is actually more accurate than the traditional law of cosines for spherical trigonometry. In order to obtain the final result for this formula, inverse trigonometry is used.

2.3.1 GPS positioning
The Global Positioning System (GPS) [12] uses the values measured by longitude and latitude as points on a coordinates net. GPS is controlled by The US Department of Defense which controls over 24 satellites orbiting our planet. The encoded signals from the satellites are broadcast to GPS receivers which then decode them. Based on these signals it is possible to determine altitude, latitude and longitude. Countries other than the USA are developing their own systems such as the GLONASS (Russian) [12] or the Galileo (EU) [12]. The GPS system is a very accurate system but it can be expensive and the receivers consume a large amount of battery power.

2.3.2 Positioning using Cell-ID
Mobile GSM devices possess identifiers named a Cell-ID [13], which is the identity of the cell that the phone is currently in. This identifier can be used for positioning the mobile device with the assistance of a Base Transceiver Station (BTS) [13] that the device is communicating with.
This method could be quite inaccurate depending on the position of the mobile device.

2.3.3 Mobile Triangulation
The Cell-ID positioning method can be improved with the assistance of mobile triangulation [14] in order to increase the accuracy. This method depends on the concentration of base transceiver stations, which implies a higher accuracy in urban areas as these usually have a higher concentration of base transceiver stations. The process of triangulation is a well known process that measures angles. Figure 2.2 shows how this system works.

![Figure 2.2 Triangulation](image)

Figure 2.2 shows the required green distance D, the blue distance L and the green angles a and b. These four elements provide the formula $L = \frac{D}{\tan a} + \frac{D}{\tan b}$ from which $D = \frac{L}{\left(\frac{1}{\tan a} + \frac{1}{\tan b}\right)}$.

2.3.4 Assisted Global Positioning
Assisted Global Positioning (AGPS) [13] is when a base transceiver station has a GPS receiver and with the assistance of the mobile network and Cell-ID positioning, a mobile device can use the GPS data to position itself. This is very helpful if the device is used in non optimal environment settings.

2.3.5 Mobile positioning
JME uses the package `javax.microedition.location` for location-based services [15]. This package provides the applications with the class `LocationProvider` based on the parameters from the class `Criteria`.

The hardware platform must have at least one location method in order for positioning to be possible. The class `Criteria` has methods for setting
the desired options that the hardware positioning system should meet. Once the options in Criteria are met it is possible to retain a LocationProvider, either to use a method to set a single position or to implement a listener to the class that regularly fires location based events. The Android system uses a similar package [16] for the location system, the difference being that the location manager has to be set in the application using the middleware.

2.4 3G
The Third Generation mobile network [17] is a standard for a wireless network that has been developed in order to replace the older 2G standard. The main difference between 3G and 2G is that 3G has been defined for higher data rates.

2.5 P2P networking
Peer to Peer (P2P) Networks [2] is different from the traditional Client-Server architecture because every node on a P2P network acts as both a server and a client. This means that each node on a P2P network is equal to any other. Gnutella [18] is a P2P system that falls under the General Public License (GPL). Gnutella has the key feature that many different applications can access the system, and is based on a tree system where the nodes are passing messages to each other through the network tree. In this thesis the nodes are called Servants as is the case in Gnutella.

2.6 Context aware applications
Context Aware Applications [19] can be described as applications that are aware of a user and the user’s environment. The mobile devices with the possibility of positioning the user are an idealistic tool for developing these types of applications. The Tallahassee Memorial Healthcare [20] uses context aware applications in the form of a location based service for asset tracking and these services has been used since 2006. Another example of usage of context aware applications is an airport that uses the positioning of security personal who are in the vicinity [20].
3 Methodology

In order to achieve the goal for this thesis, some theoretical studies are initially required within the field of geographical positioning, mobile networks, and P2P network architecture and context aware applications.

In order to achieve the desired results for the low level problems formulated in part 1.4 the following methods are to be used.

1. How is it possible to accomplish the middleware to be self organized, stable and have the possibility to fall back on a central approach, particularly if there are no servants to interact with?

**Method:** A variant of Gnutella [18] will be used for the P2P middleware and for the central approach a central server will be created. For practical reasons a simulation system to test the self-organization of the location-based network is required, for example to test when many servants connect and disconnect.

2. How is it possible to determine the position of the servants so as to determine whether the servants fall within the same vicinity?

**Method:** The usage of real GPS system should be utilized where possible. When testing the GPS system, it is necessary to use spherical trigonometry in order to find other users. Practical tests of the location-based system are going to be performed, in order to act as a proof-of-concept and to check the required quality of service.

3. Is it possible to set lower end connection of this middleware to a TCP socket type?

**Method:** Measurements are required, such as the round trip time in milliseconds for the packets to travel between different kinds of peers.

4. How is it possible to design the P2P network for different JVM [3] platforms?

**Method:** The implementation of the location based system will use the Java programming language and different kinds of Java platforms, such as JSE, Android and JME.
a. JSE is for use with a Java platform on a laptop.


c. JME uses the Java platform on a Sony Ericsson C702 [22].

5. How is it possible to implement the necessary Java code?

Method: The object oriented language Java has many useful classes and methods. Exception handling will be one of the most important parts of the Java code with the try, catch and, finally, block. There is a requirement to catch and handle exceptions in different cases, for instance: the java.net.SocketException when a Socket connection fails. EventListeners will be used for the location part of the system and because of the different settings in the class Criteria the GPS system will create different kinds of events. The middleware system will be created in a manner similar to a process, with a main class that stores the necessary static variables for the middleware and with helper class threads that perform the action. Communication with the helper threads will be dealt with in different ways depending on the platform. An application will handle the middleware through an interface and it is intended to use a special class in order to form a singleton of the main class in the middleware. The interface will then implement its methods in the main class in a manner similar to stubs.
4 Design

In some parts of the text Java code could be added in order to provide a better understanding regarding the way that a solution works. Part 4.1 describes the organization of the Location Based Personal Network. Part 4.2 deals with the architecture and the mathematics when designing a self-organizing P2P network. Part 4.3 deals with the required mathematics to enable positioning. Parts 4.4 to 4.8 describe the program design of the P2P system. Appendix A.1 has a link to the source code.

4.1 The architecture of the P2P system

The main architecture design of the P2P system is a hybrid variant as shown in Figure 4.1.
The central server registers new servants and provides them with the IP address of a compatible servant already registered. If the previous servant is disconnected and therefore unreachable, the central server is contacted again for another servant. In the process the central server removes the disconnected servants. The registry should, as far as possible, only contain active servants. The central server also provides the servants with information about the determined size of the P2P net and the maximum level of branches in the servant system. However, in relation to the amount of spawning, the servant-tree has already been provided as a constant within the architecture.

It is then up to the servants in the P2P net to connect to each other to inform the other servants about their IP addresses and their actual coordinates. The servants recognize the level of the Time To Live (TTL - how many times a message should redistribute in the system) as:

1. If the TTL has not been changed then it is a neighboring servant that is connecting, and then the actual servant must add to the internal ip list with the address of that servant. The actual servant also checks the coordinates of the contacting servant to determine whether it is in its vicinity.

2. If the TTL shows that a hop has been conducted, then the servant checks the coordinates to determine whether it falls within its vicinity.

If the given geographical area, measured by the values of the GPS coordinates, matches, then the servant with the compatible coordinates is connected and the IP address of that servant is added to a collection of type Hashtable. The connected servant also adds the IP address of the connecting servant to its Hashtable. The content of the Hashtable is then used by an application in order to interact with the servants within the given geographical area.

4.2 P2P-net architecture

The model for the P2P-net has been Gnutella [18]. Figure 4.2 shows more exactly how the P2P-net is organized. The servant tree has three levels and is spawned in a three way system, which means that each servant has the ability to connect to a maximum of three servants.
The system is organized in levels and the maximum number of servants grows with the levels as according to the formula:

\[ 3(2^n) - 2 = m \]

where \( n \) stands for the highest level and \( m \) stands for the maximum number of servants. This formula works for a spawning factor of three. For practical reasons the spawning is limited to three, because of the mobile devices limited connectivity.

![Diagram](image)

**Figure 4.2** The architecture of the P2P-system, the arrows directions indicates that the servant is to connect with the servant that is pointed on.

When there is a given number of servants it is possible to calculate the level of the servant tree by rearranging the formula in 1. as:

\[
 n = \frac{\ln \left( \frac{m + 2}{3} \right)}{\ln 2}
\]

where \( n \) is once again the level and \( m \) is the number of servants. The reason why the base E is used is because Java uses only the natural logarithm in the class Java.Math. Observe that the value \( n \) is of the datatype double; therefore proper rounding to the nearest higher integer
has to be performed. When an exact value of an integer has been passed then this indicates the start of a new level. As an example:

Servant A4 in Figure 4.2 level = \( \ln \left( \frac{(4+2)}{3} \right) / \ln 2 = \ln 2 / \ln 2 = 1 \)
Servant A5 same Figure = \( \ln \left( \frac{(5+2)}{3} \right) / \ln 2 = \ln(7/3)/\ln 2 \approx 1.22 \)
Observe that the level starts from zero as in a Java array.

For the calculation of a proper servant, the following formula is used:

\[
\frac{(I_2 - 2)}{2} = I_1
\]

where \( I_2 \) is a servant in the lower level and \( I_1 \) is a servant in a higher level. The variables \( I_1 \) and \( I_2 \) are the servants’ index numbers in the Java collection and the reason for using the index collection number is that it is easier to find the servant directly by using its collection number. Since the servant tree is populated in the direction from right to left, the numbering in the collection works in a similar manner. As an example:

See the servants from Figure 4.2, but this time use servants A5 and A6. These servants have collection numbers 4 and 5 respectively. By using the formula in 3 this provides the following:

\[
\begin{align*}
A5 & \quad \frac{(4 - 2)}{2} = 1 \\
A6 & \quad \frac{(5 - 2)}{2} = 1.5
\end{align*}
\]

If required, the result is then rounded to the nearest lower integer and for servants A5 and A6, the servant with the collection index number 1 and identity A2 is given.

For the servants A2 –A4 the result is lower than 1, and therefore the servant with the index 0 is used, in this case servant A1.

From Figures 4.3 and 4.4 some examples of changes when a servant disconnects can be seen. Because of the manner in which the servant tree is populated, the collection-number is always the same for a position. When a servant is disconnected the result is that a “hole” in the P2P net appears and the last attached free servant is used to fill in that gap. In this case the term “free” means a servant that it is not connected by another servant.

So the servant A10 in Figure 4.3 receives number 1 automatically and the servant A10 in Figure 4.4 receives the number 0.
The type of collection used is a List from which it is possible to either obtain an Object or a collection number as an index of an Object.

As in other P2P systems the servants contain both a socket and a serversocket, and this is the reason that Gnutella calls the servant Servants.

Figure 4.3 the servant A2 has been disconnected and the last free connectible servant A10 has taken its place.

Figure 4.4 the servant A1 has been disconnected and the last free connectible servant A10 has taken its place.
4.3 Geographical Coordinates

From the theory part, it is known that the Earth is not a perfect sphere, so in order to obtain the actual radius for a certain position, the following formula is used:

1. Sphere radius = 6378.1 – 21 x sin (latitude)

Formula 1 is used for the author’s own location (Dalgatan in Sundsvall of the country Sweden) where the latitude is:

2. Latitude = 62.3911911

3. Sphere radius = 6378.1 – 21 x sin (62.3911911) ~ 6359.5

In order to calculate and test distance to appropriate servants, the Haversine formula is implemented as a method using the Java code, see Figure 4.5 in order to see how it is used.

```java
public boolean distance(double longitude1, double latitude1, double testdistance)
{
    double londiff = longitude2 - mylongitude;
    double latdiff = latitude2 - mylatitude;
    double a = Math.pow(Math.sin(Math.toRadians(londiff/2)),2) +
                Math.pow(Math.cos(Math.toRadians(mylatitude)) * Math.cos(Math.toRadians(latitude2)) *
                Math.pow(Math.sin(Math.toRadians(londiff/2)),2),2);
    double c = 2 * Math.atan2(Math.min(1, Math.sqrt(a)), Math.min(1, Math.sqrt(1-a)));
    double distance = 6378.1 – 21 * Math.sin(Math.toRadians(mylatitude)) * c;
    if(distance<testdistance)
    
    return true;

    return false;
}
```

Figure 4.5 Implementation of Haversine’s formula on Android SDK.
When using the method from Figure 4.5 a P2P network can be formed with all the servants within the ring described in Figure 4.6.

Figure 4.6 Example of the connection area that the servants could reside in. This Figure was made with the help of [23]
4.4 Central Server

The central server and registry have been written on a JavaEE5 platform, and the system is running on an Apache Tomcat 6.0.18 Server.

The UML for the system is in A.2.

The application contains the following classes:

1. HawkEyeFive.java, which is a generic of the type:

   ```java
   public class HawkEyeFive<T1, T2, T3, T4, T5>
   ```

   This generic class makes the handling of the registration the servants easier. It is very useful when stored in a List of the type: `ArrayList< HawkEyeFive >`, this list is used by the servlets when the servants registered are required to be shown.

2. index.jsp: a start file for the web part where it is possible to choose to link to the central server system. Index.jsp is a file of the type Java Server Pages, which means server scripting.

3. MultiServerStart.java: a servlet that starts the central server, which is a thread. This servlet also shows all the servants that are registered.

4. Central_Server.java: is a Java class thread that handles the serversocket and that accepts connections in order to allow a helper thread to handle them. The resulting socket from an accepted connection is sent to the helper threads constructor.

5. Central_Server_Helper.java: is a Java class thread that handles connections accepted by 4. In this thread’s constructor there is a socket sent by 4. that creates DataInput/OutputStream for byte streams. These streams are used for communication with a client.

6. Xmlmaker: is a thread that is started by 4. This thread creates a log file in the form of an XML-document and the DOM document is constructed by the thread with the assistance of the class document-builderfactory. This is the method that handles the DOM, and a node with processing instructions is added to the
document in order to create an XSL transformation (XSLT) on the XML document.

4.5 The P2P system for JME

An important part of the JME version is the class StringItemHandler that extends String-Item. StringItemHandler is used as a handler for the threads so that they can send status messages to an application. When the main class for the system is instanced, as in a singleton, there is a forcing moment to form the necessary objects in the heap otherwise exceptions will be thrown. Even though the system is self-supporting it is always interesting to evaluate the actual state of the system. In order to show that the system is separated, there are special classes that handle exceptions.

The UML for the system is in A.3.

1. P2P_Connection_Interface.java: is an interface that an application uses to interact with the P2P system. This interface has a number of methods that can be used by the application, some of which the application must use in order to start the P2P system and others are optional for the user, based on the desired service.

2. Uppgift_Network_Connection.java: a final class that acts as the main class even though it does not have a static main part. This class implements 1. and the collaboration between 1. and 2. And can be compared to a stub system. This is the only ongoing class that is not a thread and all the static instance variables used by the threads are stored here. All the necessary threads are constructed and initiated here in the connect () method.

3. P2P_Connector_Instance.java: this final class contains a static method where an application creates an instance of 2., which is comparable to forming a singleton (exactly one object) and returns the interface 1. The arguments for the static method getP2p(…) are the protocol-name, the host-address for the central-server, the port that the central server uses, a number of StringItemHandlers and a Hashtable. A special exception class named P2pConnectorException is used only for this class, in order to throw new exceptions for an application to catch and observe. Exceptions are thrown in a number of cases: a. the
protocol is malformed. b. the String’s host address and port are empty or malformed. c. StringItemHandler objects are not created and therefore have a nullpointer. d. The Hashtable that the application uses is not allocated in the heap and therefore a NullPointerException is thrown.

4. Central_Server_Handler.java: a thread that is started by 2. and handles the registration of and querying for an active servant on the central server. The communication with the central server results in an active IP address that is used by the P2P system.

5. P2P_In_Server.java: this thread is started by 2. and is the server part of the servant. A SocketConnection is created when the ServerSocketConnection’s acceptAndOpen() is called, the SocketConnection is passed to a constructor in a Server-HelperThread. This thread never sleeps and thus the ServerSocketConnection must be open, and can only be stopped by calling interrupt(). This thread’s ServerSocketConnection is used both by the application’s location based P2P network and the internal P2P locator system that handles the internal IP addresses.

6. Locations_Cordinates_Provider.java: this thread provides the system with the GPS coordinates namely the latitude and longitude. This coordinates are then used by the system in order to create local P2P networks. This thread has a LocationListener that listens to events started by the GPS system, when a new position is received. The method locationupdated(...) is used by the event listener in order to create the new Coordinates. The thread sleeps for 1000 ms and then updates the static double variables latitude and longitude.

7. P2P_In_Serverhelper.java: is a thread that receives SocketConnections from 5. and creates a DataInputStream/OutputStream for byte streams. The byte packages received are checked by 2. in order to determine a location.

8. P2P_Out_Socket.java: is a thread that is the socket (client) part of the servant. This thread’s socket is used by an application in order to send messages to other servants that are part of an application’s location based P2P network.
9. P2P_Location_Checker: this important thread handles the internal IP address system. This thread sleeps as long as the variable integer interval allows. The interval variable is set relative to the level of the servant tree (number of servants) of the P2P system, because the more servants there are the longer the required sleep with reference to the propagation time of a sent message. This thread connects the servant obtained from the central server and in case the servant is not active, a new request is made with the help of 4. in order to obtain an active servant. The sent message contains the original IP address, sender IP address the Coordinates provided by 6. and a TTL based on the level of the servant tree. Observe that 5. handles the incoming messages from other servants P2P_Location_Checker’s SocketConnections.

4.6 The P2P-system for Android

The Android version of the P2P system has many similarities when compared to the P2P system described in 4.5. The parts of the Java code that are special for Android will now be described in greater detail. The class Handler is a very useful class when the status of the threads is required to be seen. A Message is created in a thread by the method handler.obtainMessage(), which obtains a new Message from the global message pool. After a Message is created a Bundle is created which can hold a number of classes. As an example the method putString-ArrayList(String key,ArrayList<String> value) can be used.

The handler in an application, or in a position in which the Messages are to be seen, obtains the Message as getData().getStringArrayList(key). Observe that the key was set by the thread.

The UML for the system is in A.5.

1. P2P_Connection_Interface.java: is similar to the one in 4.5 1.
2. Uppgift_Network_Connection.java: it was required that the Collections for use were typed in a safe manner and, additionally, the StringItemHandlers have been replaced by Handlers.
3. P2P_Connector_Instance.java: As in 2. the StringItemHandlers has been replaced by Handlers and a LocationManager must be created by the application.
4. Central_Server_Handler.java: Uses Sockets instead of Socket-Connections as in 4.5 4. and has a method messenger to create Messages as has been described previously.

5. P2P_In_Server.java: Uses ServerSocket instead of ServerSocket-Connections as in 4.5 5. and has a method messenger to create Messages as previously described.

6. Locations_Cordinates_Provider.java: There are fewer methods to the class Criteria in the constructor of this thread as compared to 4.5.6.

7. P2P_In_Serverhelper.java: is a thread that receives Socket-Connections from 5. and creates a DataInput/OutputStream for byte streams.


### 4.7 The P2P-system for JSE

The JSE version of the P2P system has many similarities when compared to the P2P system described in 4.5. and uses the same type of low end TCP connections as in 4.6.

The main difference is that the P2P system for JSE does not use Handlers as described in 4.6, but rather uses javax.swing components and is forced to send them to this system by the instantiation (ref 4.5 3. and 4.6 3.). This component’s textual content is used by an application to view the status of the P2P system. In the case of serious exceptions, such as a central server being offline, the system informs the user through a JOptionPane.showMessageDialog() and then shuts down with System.exit(1). The geographical location is hardcoded in the JSE version, because of the availability of GPS hardware.

The UML for the system is in A.4.
4.8 P2P simulation system
The P2P simulation system interacts with the Central Server in the same way as in 4.5, 4.6 and 4.7 in order to simulate a large P2P net of servants.

The UML for the system is in A.6.

1. Peer_to_peer_filler.java is a thread that fills the servant tree with servants by using fictitious IP addresses of the type aN or fictitious IP addresses given by a user. The thread can fill the tree with as many servants as is required.

2. Peer_to_peer_filler_main.java is a class that extends JFrame and implements ActionListener. The window that this class creates is used to receive fictitious IP addresses that a user types. If no IP addresses are typed then 1. creates a series of aN IP addresses. This class also starts 1.

3. Peer_to_peer_removal.java is a thread that removes servants with the IP address given by a user. The constructor for this thread takes three arguments: a. The IP address that could not be connected b. The IP address that is attempting to connect to a. c. The last free reachable servant.

4. Peer_to_peer_removal_main.java is a class that extends JFrame and implements ActionListener. The window that this class creates is used to receive IP addresses that a user types. These IP addresses are sent to 3. This class also starts 3.
5 Results

The thesis results are presented in this chapter.

5.1 Application for JME

The application for the JME platform is used by Sony Ericsson C702 as seen in Figure 5.1

![Application using the P2P system on Sony Ericsson](image_url)

Figure 5.1 Application using the P2P system on Sony Ericsson

The application works as a chat program between the servants that are in the location ring. Firstly, there is a TextField in which the user can type messages that can be seen by all the users in the vicinity. Then it displays status information about the Central Server, the server part of the phone and the actual coordinates that the user resides in. In the options part a list of servants that are in the vicinity can be seen.
5.2 Application for Android

The application for the Android platform is used by HTC HERO as shown in Figure 5.2

![Figure 5.2 Application using the P2P system on HTC HERO.]

The main function of this application is the same as described in part 5.1. Android has a significant advantage in that the layout system is handled in separate XML files. The background layout uses a shape of the type gradient and when an event of any kind has occurred (ex. receiving a message), a MediaPlayer plays a short sound. If a user wants to see the servants within the location ring, the menu key should be pressed and then a list of servants becomes visible. For practical reasons only portrait mode is used and the back key is restricted.
5.3 Application for JSE

The application for the JSE platform is used on a laptop can be viewed in Figure 5.3

![Application using the P2P system on my laptop.](image)

The main function of this application is the same as described in part 5.1. The application uses only javax.swing components for the layout and the geographical coordinates are hardcoded.

5.4 Telia 3G

Unfortunately the IP addresses starting with 90.237.x.x are unable to form a connection with each other, so the testing of a net with multiple mobile devices could not be accomplished in Telia 3G. When an attempt was made to obtain a socket connection with another mobile device in the Telia 3G Net a sockettimeout exception was thrown, regardless of the given value of sockettimeout.

This problem is most likely a NAT traversal problem and therefore Telia is required to solve this, so that mobile servants can connect.

The Android mobile device that was tested had the same problem as mentioned above, but when a new WIFI connection was created, the socket connections worked without any problems.

The result is therefore, currently, that servants using Telia Mobile 3G are unable to perform a socket connection to each other.
5.5 P2P net simulation

It proved to be impossible to construct a large P2P net of mobile devices, referring to the fact that in 5.4, the choice was made to simulate a P2P net with a starting of 36 servants in accordance with Figure 5.4.

The reason why the choice to simulate a larger P2P net has been made is in order to demonstrate that the P2P’s tree structure is self-organizing and consistent.

For practical reasons, the servants IP addresses αN has been used instead of real IP addresses.

In Figure 5.4 it is obvious that free and reachable servants have zero bindings, which means no arrows pointing from other servants. The servants used for relocation always had zero bindings.

![Figure 5.4 P2P net with 36 servants.](image-url)
The test was performed making the following rearrangements:

1. Topservant A1 with ordernr 0 disconnected and replaced by servant A36, because A36 is the last free connectible servant.

2. Servant A15 ordernr 14 disconnected and replaced by servant A35.

3. Servant A9 order nr 8 disconnected and replaced by servant A20. Because servant A20 is the last connectible servant, a recursive search had to be conducted and servants unable to be connected were removed.

Figure 5.5 shows a screenshot from the command screen with the actual positioning after the rearrangements above.
Figure 5.5 showing the positions in the P2P net after rearrangements.

Connectnode is eg the connect servant. Bindings mean the number of servants that connect to the actual servant in the IP address. Connectnode is the servant that the IP address connects to. Level means the tree level of the servant tree.

After performing this simulation in the P2P net, the result found is that the P2P net is self organizing.
5.6 Evaluation of Location Based Math

For the Android mobile device the method based on the Haversine formula is tested. In order to obtain a visual representation of the result of the formula, the test was made in NetBeans-JSE, but the same formula is used in Android. Figure 5.6 shows a location outside the location circle.

Figure 5.6 showing a location outside the location circle.
Figure 5.7 Screen shot from the test on Netbeans Ide.

Figure 5.7 shows the result when using the Haversine formula. The testing coordinates is compared with the coordinates of the author and the result is the distance calculated to approximately 1.1172 km which means that the testing coordinates are outside the circle.
5.7 Location Based Network Evaluation of Connection Roundtrips

As the devices were connected to the central server, the time in milliseconds was measured, used by the central server for the roundtrip to write and read UTF to the mobile units. Tables 1 and 2 show the acquired values. It should be noted that the Android unit in Table 2 is connected to a local WIFI –point.

Table 1: Connection results for JME.

<table>
<thead>
<tr>
<th>Sony Ericsson C702 JME</th>
<th>Try Number</th>
<th>Round trip delay ms</th>
<th>Value x 0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>399</td>
<td>199.5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>717</td>
<td>368.5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>757</td>
<td>378.5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>897</td>
<td>448.5</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>404</td>
<td>202</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>778</td>
<td>369</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>412</td>
<td>206</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>779</td>
<td>389.5</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>730</td>
<td>366</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>390</td>
<td>199</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td><strong>313.55</strong></td>
</tr>
</tbody>
</table>

Table 2: Connection results for Android SDK.

<table>
<thead>
<tr>
<th>HTC Hero Android SDK</th>
<th>Try Number</th>
<th>Round trip delay ms</th>
<th>Value x 0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>210</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>216</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>438</td>
<td>219</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>211</td>
<td>105.5</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>210</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>203</td>
<td>101.5</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>203</td>
<td>101.5</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>206</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>206</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>320</td>
<td>160</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td><strong>121.15</strong></td>
</tr>
</tbody>
</table>
5.8 Location Criteria

In order to better control the LocationProvider a class Criteria is used and the object that is created takes a number of arguments for its methods. One of the more interesting is the method `setCostAllowed(boolean)` for JME, because the argument true must always be used otherwise a nullpointer exception is thrown and the location system will not be initiated.

The method `setPreferredPowerConsumption(…)` works very well with the argument final int POWER_USAGE_LOW, because when using other final variables as arguments the mobile devices became very hot, but in this case the devices did not become nearly as hot.

5.9 Java.Math and JME case 1

Unfortunately JME do not have arcsines in their java.lang.Math. Tests were made with MathFP [24] as shown in Figure 5.8

```java
public boolean distance(double longitude2, double latitude2, double testdistance) {
    double londiff = longitude2 - mylongitude;
    double latdiff = latitude2 - mylatitude;
    double aa = Math.sin(Math.toRadians(londiff / 2)) * Math.sin(Math.toRadians(latdiff / 2));
    Math.cos(Math.toRadians(mylatitude)) * Math.cos(Math.toRadians(latitude2)) * Math.sin(Math.toRadians(londiff / 2));
    double sqrtjme=Math.sqrt(aa);
    int sqrtjme= (int) MathFP.toFP(sqrtjme);
    System.out.println("sqrtjme: "+ sqrtjme+" sqrtjme in string: "+MathFP.toString(sqrtjme));
}
```

Figure 5.8 Observe the rectangle.
From the result in Figure 5.10 and the marked rectangle it is obvious that MathFP is unable to handle small values because the original value is $8.16443126 \times 10^{-5}$ and MathFP interprets that value as double 8.1643. So if MathFP is used, the value in the method MathFP’s arcsines will obtain incorrect values.

The result is that the Haversine’s formula and other formulas using inverse trigonometry are unable to be used in the mobile system JME.

### 5.10 Java.Math and JME case 2

In order to mathematically determine the level of the P2P servant tree a variant of the formula described previously is required:

$$n = \ln\left(\frac{m}{3+1}\right) / \ln2$$

The formula is, in this case, particularly for the collection index number as opposed to the earlier formula in 4.2 that managed the number of servants or the size of collections.

This formula is important when a decision is required in relation to the means of handling the servants depending on the actual tree level that the servants reside in. However, the JME’s Java.Math class is unable to perform the calculation \(\log(\text{double})\) and that is a problem when calculating natural logarithms.

The MathFP [24] has a class that should solve this problem but unfortunately it was too inaccurate as shown in Figure 5.9.
Figure 5.9 Observe the rectangles.

The correct answers to the calculations in Figure 5.9 should be:

1. for $7.0 \ln \left( \frac{7}{3} + 1 \right) / \ln (2) \approx 1.7369$
2. for $8.0 \ln \left( \frac{8}{3} + 1 \right) / \ln (2) \approx 1.8744$
3. for $9.0 \ln \left( \frac{9}{3} + 1 \right) / \ln (2) = \ln (4) / \ln (2) = 2$

This inaccurate behavior makes it impossible to determine when a new level has been reached.

The result is that JME cannot be used for the calculations in 4.2 2.
5.11 Memory weakness in Sony Ericsson C702
In spite of the fact that the P2P system uses only lightweight processes as threads, the JME platform on Sony Ericsson throws Memory Extension Exceptions if there are more than two servants. The Android device HTC Hero had no such problems even with three servants, which means one JME emulator, one JSE application and the Android phone.

5.12 P2P System Field Evaluation
Field testing of the location based P2P system has been conducted, and the results can be viewed in Figures 5.10 – 5.11.

Figure 5.10 Fieldtrip testing traveling by bus observe the small ellipse.
Figure 5.11 Fieldtrip testing done by walking observe the small ellipse

The field testing was performed on a JME Sony Ericsson C702 and Android HTC HERO. In order to form a P2P net the Android device used a different 3G distributor to Telia 3G because of the fact in 5.4. The WIFI point created for Android in 5.4 was too local and could not be used outside the building. The small grey ellipsoids in Figure 5.10 and 5.11 show the error margin. The accuracy settings in Criteria were 5m in both cases. The JME device was used as a non locating device because of the facts in 5.9, meaning that the Android device could perform locating servants using the JME device.

The result is that the servant outside the rings of Figure 5.10 and 5.11 is not connectable and when the user was inside the rings the servant was connectible.
6 Conclusions

6.1 Low level problem formulations

This section starts with a review of the questions from part 1.4.

1. How is it possible to accomplish the middleware to be self organized, stable and have the possibility to fall back on a central approach, especially if there are no servants to interact with?

2. How is it possible to determine position of the servants in order to determine whether the servants are in close vicinity?

3. Is it possible to set lower end connection of this middleware to a TCP socket type?

4. How is it possible to design the P2P network for different JVM platforms?

**Question 1** has been proved by the simulations in part 5.5 for larger systems and could be applied to Android and JSE platforms if they use other 3G distributor than that described in part 5.4. However, for JME platform the stable condition fails because of the facts stated in part 5.11 for the tested device and also for the calculation fact states in part 5.10 for all JME platforms.

**Question 2** has been proved by calculations in part 5.6 and field testing in part 5.12 for Android systems. For the JSE system it was not possible to determine locations because ordinary laptops do not use GPS systems. Because of the limitations in part 5.9 for JME platforms and 5.11 for the device that was tested, location based systems cannot be used on Sony Ericsson C702.

**Question 3** has been proved for Android and JSE platforms if a 3G distributor other than that shown in part 5.4 is used. Even if a different 3G distributor is used, the limitations shown in part 5.11 show that it is not appropriate to use TCP connections in a large scale system on Sony Ericsson C702.
Question 4: The P2P system has been designed as shown in part 4 and tested positively for both Android and JSE platforms. The P2P system has also been tested on JME, but the tested device had memory limitations.

6.2 High level problem formulations

The question from part 1.2 was:

Is it possible to find a technical solution in the form of a middleware that enables location-based, lightweight and self-organizing P2P networking between different kinds of servants?

The answer is yes for Android and JSE platforms, with reservation for the 3G distributor used, and as the situation is at present, it is not possible for JME devices.

6.3 Discovered problems

Here are the problems that were encountered while accomplishing this thesis.

The emulator (SonyEricsson_JP8_240X320_Emu) and the real mobile device (Sony Ericsson C 702) had very different layouts and this caused problems when attempting to create a layout for the real mobile device. The Command items for example are placed in reverse as compared to the real phone.

The battery power of the mobile devices does not last as long as desired, when testing the mobile devices. A maximum of 4 hours has been achieved.

When implementing the P2P system for JME it is important to write code that checks whether the Displayable are active or not, otherwise the device shuts down without an exception.

The location listener in the JME device sometimes takes a considerable time to obtain a location from the location provider.

The server part of the P2P system on the JME device cannot be trusted to shut down with interrupt(), because, on occasions, the ServerSocketConnection is still open and the system cannot be restarted. This problem appeared randomly and had nothing to do with the
workload of the device. The only solution was to wait for the thread to finish.

Every time the JME device was started, using the 3G distributor as described in part 5.4, a new IP address was obtained. In order to maintain the P2P self organizing system consistent, the old IP address had to be stored in the JME device with the help of RecordStore.

The Android Qwerty keyboard is difficult to handle for a standard sized person in relation to typing words.

6.4 Future Works

It is really necessary to evolve the JME java.langMath part, as shown in the result parts 5.9 and 5.10 because the P2P systems are based on several mathematical theorems. It is also imperative to redesign the memory system of Sony Ericsson C702.

Another aspect which is missing in JME involves binary streams for objects.

Studies have been made in order to enhance the location based P2P system, and one interesting idea is to use ultra peers, which are also used in later versions of Gnutella. Since many kinds of applications on many different kinds of platforms are used, it would be interesting to evaluate the collaboration between high performance devices and devices that are more limited with regards to battery time and computational power.
References


Appendix A: Documentation of own developed program code

A.1  http://portal.miun.se/~joni9900/P2PContextAware.zip
A.2  http://portal.miun.se/~joni9900/ServerProjektA.jpg
A.3  http://portal.miun.se/~joni9900/ErikssonUmlA.jpg
A.4  http://portal.miun.se/~joni9900/P2PJseUmlA.jpg
A.5  http://portal.miun.se/~joni9900/AndroidUmlA.jpg
A.6  http://portal.miun.se/~joni9900/FillerA.jpg