Software technology project

Location prediction with Android devices

June 17, 2011

Group number: N1

Navn: Peter Gammelgaard Poulsen (s093263)
Johan van Beusekom (s093251)
Lars Libak Pedersen (s093278)

Danmarks Tekniske Universitet

Abstract

This report covers the development of a software project concerning prediction of peoples movement patterns. This is done by tracking people's position via their Android phone, for which we have developed an application which periodically sends the location of the user to a database. The database then uses the collected data to try and make a prediction of the users position, using an autoregressive model, which is one of several methods of predicting various types of natural phenomena.

The Android application and the webservice are developed in JAVA, while the prediction algorithm is coded in MatLab. The development of the mobile application, the set up of the database and server, as well as the making of the prediction engine are all covered in the report.

Contents

1 Introduction

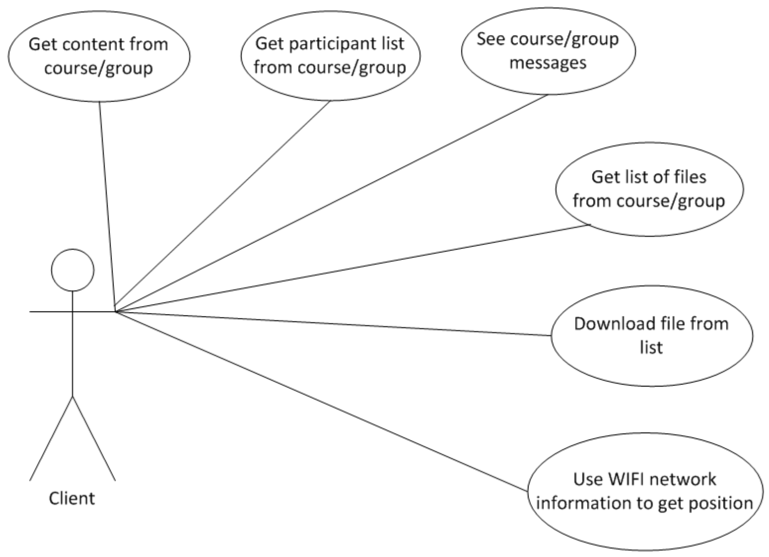
The question of whether or not a person's movement pattern is predictable is interesting, mainly because there is no clear answer of this and because it may vary a lot from person to person. Would the knowledge of us being predictable, if that is the case, change the places we go, and the things we do, to make ourselves less predictable, less observable and strengthen our belief in the fact that we do what we do by our own free will? Are certain groups of people more predictable than others, say for example are DTU students more predictable than KU students, because they have a campus to go to and a lot more lectures? Do the patterns of the people you are around during your everyday life change the way you behave and move? These are all questions that are tough to answer, because there is no model of deciding this easily. In fact what you need is a lot of data and the use of empirical research, which means learning from that data, in order to create a model describing your system. When it comes to making a reliable prediction of a group of people or an individual, the golden rule is the more observations, the better, because by having a lot of observations it is easier to describe general tendencies and eliminating flaws caused by deviations of the group or individual.
Up until recent years making such an experiment has been close to impossible because the collection of data could not be made automatically, but with the introduction of mobile phones and other mobile devices, it is now possible to collect data such as where a phonecall was made, GPS locations, wifi connections and so on. Once it is possible to store this data with both a timestamp and a connection to a person, there are certain ways of creating models that describe patterns in the data.
In our project, we are trying to make such predictions of movement patterns as described below.

1.1 Usage

What can we learn from all the collected data? The first thing that comes to mind is of course the thing we can learn about ourselves as human beings, we can find out just how predictable people are and if some are more than others, we can find out if groups of people move together at times, for example student of the same class at a school, which might be relevant and interesting to psychologists. Of more practical usages the user could use the data to track his friends, and find out where they are most likely located in the future. For an institution like DTU, the information about which buildings that are being occupied by people at certain times, might be interesting, because such knowledge would allow DTU to turn off the heat in buildings that are not being used etc.

1.2 Current standpoint and the goal of the project

In an earlier project we created an application for Android phones which basically was a CampusNet app, with the ability to get the newest messages and files from CampusNet as well as getting the same information from all groups the user is in, along with other basic functions. A use-case diagram of this is shown here:


Get content from course/group
Get participant list from course/group
See course/group messages
Get list of files from course/group
Download file from list
Use WIFI network information to get position
Client

The idea is, that we are going to extend this application to periodically send the position of the user, based on WIFI-locations around DTU, to a database, which in turn can be used for creating predictions of the users position. In the project we have been working with Android programming, matlab for creating the models of determining position, set up of a server and database as well as the communication between this and the device. Putting all this together we should by the end of the project be able to show a prototype of a distributed system which can track friends positions on a map and predict future locations based on data.

1.3 Collaboration

Our project is part of a collaboration between several other projects at DTU, that have different aims, but are all based on mobile applications, some of which use the same basic application that we developed earlier. The idea is that the work done by each project can later be implemented into the same application. Since several of the projects involved in the collaboration needed to use a remote database, we decided to all use the same instead of setting up one for each group. This has resulted in our initial database not being used, but the work we did with it is covered in the report.

2 Requirement analysis

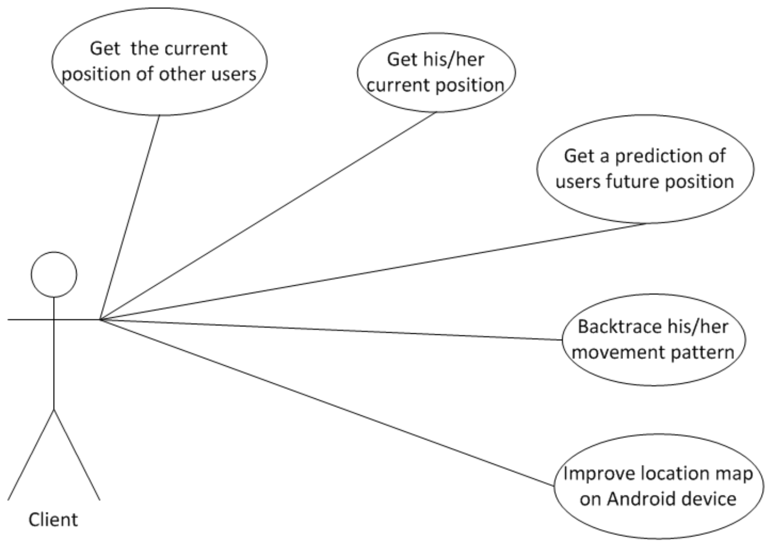
2.1 Essential terms

Here is a brief explanation of the terms we are using in the report.

- CampusNet API: A public restful HTTP API implemented by Arcanic. Allows users to extract data such as courses, files and messages etc. from their CampusNet account.
- Application: The application is the program running on the Android mobile phones which is coded in java using the Google Android API.
- User: The person that is logged in to the application.
- MAC Address: A MAC address is a unique identifier for a network interface.
- Google Android SDK: The software development kit provided by Google to build Android applications.
- Prediction engine: A service that uses an algorithm which will try to predict the user's position by looking at previous positions of the user.
- Device: The Android phone.
- Autoregressive model: The method of prediction that we've chosen to use for our prediction algorithm.
- Client: An application that can access a service. It can send data or requests to the service and get a response. The service is often called a server.
- Webservice: A service that can receive data and requests from a client and return a response.
- Android service: A way of making a method run in the background of an Android application, in our case it sends the location of the device.

2.2 Choices and constraints

When making the project, we had to make some choices as to which features we wanted to implement in the application, and which features that should be left out.
The implemented features are shown in the use-case diagram below:


Get the current position of other users
Get his/her current position
Get a prediction of users future position
Backtrace his/her movement pattern
Improve location map on Android device
Client

However some parts of the functions are on the proof of concept level, mainly those concerning the prediction algorithm, which is simplified in the implementation, but the theory behind the algorithm is explained in the report. When it comes to actual testing of our algorithm, this has been hard to do, because of the limited time we've collected data, and therefore we have run some tests on time series, that describe a groups position at one location.

3 Theory

Once all the necessary data is collected, it needs to be processed in order to make a prediction of a users location. There are several ways of doing this.

3.1 Autoregressive model

For our prediction algorithm, we've decided to use what is called an *autoregressive model* (AR), which is a statistical model that uses previous observations to try and predict future observations. Any signal that can be represented as an amplitude that varies with time has a corresponding frequency spectrum[1]. The frequency spectrum shows spectral densities as functions of frequencies in the frequency domain. What this means, is that each frequency in our spectral domain has an assigned value determining how "important" this frequency is for describing periodicities in our signal. This will be described further below, with examples to help understanding what is happening.

The autoregressive model is defined as

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t$$

where X_t is the prediction at time t, X_{t-p} is some known observation previous to t, $\phi_1, ..., \phi_p$ are the model parameters, ϵ_t is white noise and c is a constant. The order of the model, p, determines how many steps back the model knows, the more, the better.

The idea is that the model looks at the known observations and finds a system in these, for example if we observe a students position in relation to his school, we might find that he goes to school from monday to friday, while not being at the school during the weekend. If the AR model has access to observations of several weeks of the students' location pattern, it will learn from the data and try to fit frequencies onto it, describing periodicities, which in the example probably would result in a week being recognized as a period, meaning that it is highly likely that the person is the same place as he was a week ago. It is also likely that a day would be recognized as a period, because it is somewhat likely that a person is in the same location as he was 24 hours ago. The model parameters, $\phi_1, ..., \phi_p$, are based on the on the spectral densities found in the frequency spectrum, so for example, if a signal has a period of 10, then the frequency spectrum will have a high value at frequency $f = 0.1$, because this would give a period of:

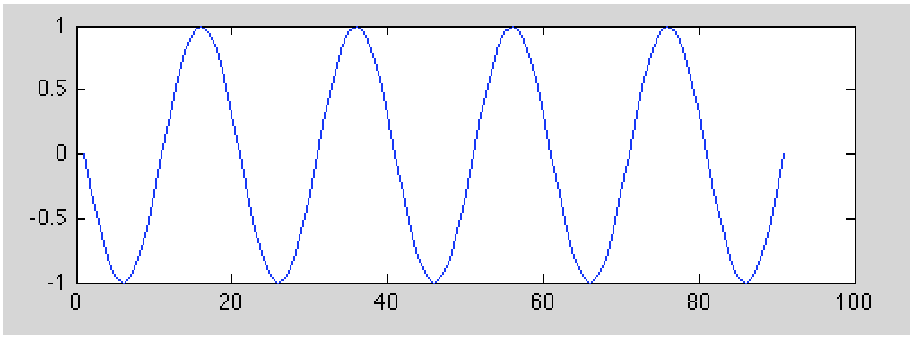
$$T = \frac{1}{0.1} = 10$$

This means that the value of ϕ_{10} would be the most significant model parameter in the AR model, which again would mean that the observation made 10 steps ago, is the most significant when it comes to predicting the next value.
To clarify things a bit, and make it easier for the reader to understand, we have some examples below to give an overview of how the model works.

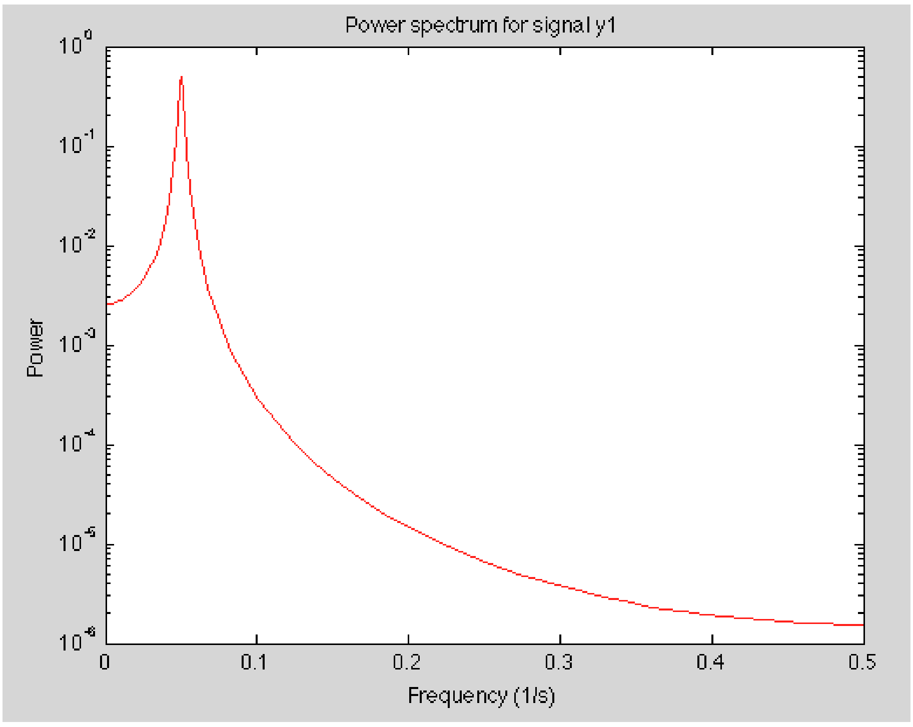
[1]Wikipedia

3.1.1 Example

Here's an example of how the *AR* model works on a sine wave. The sine wave is shown below.



The sine wave has a period of length 20 and when we apply the *AR* function from MatLab, and specify a large enough order, we get the following spectrum:



The graph shows that the frequency $f = 0.05$ is significant, which makes sense because it gives a period of $T = \frac{1}{0.05} = 20$, which is the period of our sine wave. If we have a more complex signal, which contains several periodicities (example: weeks and days), then all the frequencies describing these periodicities will be the most significant ones in the power spectrum.

3.1.2 Estimating the function

The above example, was a very simple one, because it contained only one periodicity, but if we go back to the example with a student going to school during weekdays, and being at home during weekends, we might see that our frequency spectrum will indicate several periodicities, for example one saying that there is a correlation between where you are now and where you were 24 hours ago as well as one saying that there is a correlation between where you are now and where you were a week ago.

When it comes to estimating the model parameters, there are several ways of doing this. One way of doing it is by solving the Yule-Walker equations.
The equations are defined as:

$$\gamma_m = \sum_{k=1}^{p} \phi_k \gamma_{m-k} + \sigma_\epsilon^2 \delta_{m,0}$$

where $m = 0, \ldots ,p$. γ_m is the autocorrelation function of x, σ_ϵ is the standard deviation of the input noice process and $\delta_{m,0}$ is the *Kronecker delta function*. However, the last part of the equation is non-zero only if m = 0, this part is usually left out[1], which is why we won't go into detail with this in the report. The equations can be written as matrices as shown in the figure below, which is from the Wikipedia article about Yule Walker equations.

$$\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} \gamma_0 & \gamma_{-1} & \gamma_{-2} & \cdots \\ \gamma_1 & \gamma_0 & \gamma_{-1} & \cdots \\ \gamma_2 & \gamma_1 & \gamma_0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \vdots \end{bmatrix}$$

from which the model parameters can be derived easily.

3.2 Over- and underfitting

Over- and underfitting are concepts used in statistics when trying to fit a model to a system. When we fit a model to the system, we want it to be as simple as possible, while still being correct. The reason why we want it that way, is because our data often contains a lot of noise, so while it would be possible to fit a polynomial to it, it would make no sense when using it for making predictions because the random noise would affect the prediction. This is what's called overfitting, and the opposite, making a model that is too simple to describe the system is called underfitting. Examples of over- and underfitting are given in the section *Polynomial curve fitting*.

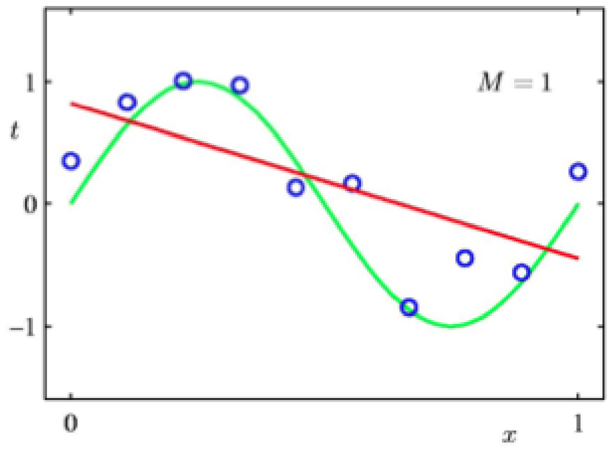
3.3 Other approaches

The autoregressive model is one of many ways of making predictions of a signal, but there is no indication that this is better than other methods of linear prediction. Another approach is using polynomial curve fitting, in which you construct a curve or mathematical function that has the best fit to a series of data points.[1] A problem with the polynomial curve fitting is that it is very likely to keep increasing or decreasing when plotting it, so it is not ideal for making predictions far into the future. Another option is the very simple method in which we define which time steps are relevant when it comes to predicting the next, for example if we are measuring a student, we can tell the model to base its prediction on the students' location one week ago.

[1]Wikipedia
[1]Wikipedia

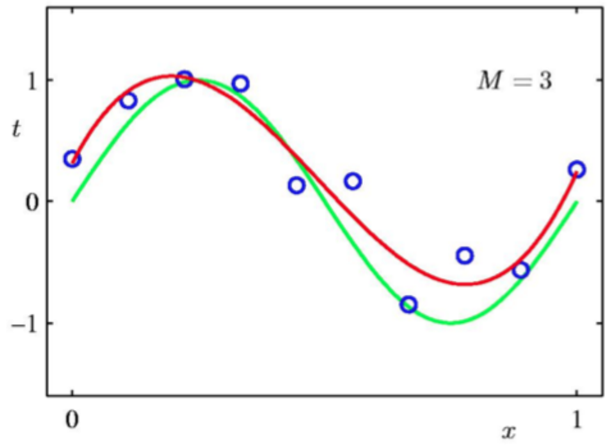
3.4 Polynomial curve fitting

Here is an example visualizing the concepts of over- and underfitting of a polynomial curve on a set of data.[2]
The first example shows underfitting, in which the model is trying to fit a 1st order polynomial to the data.



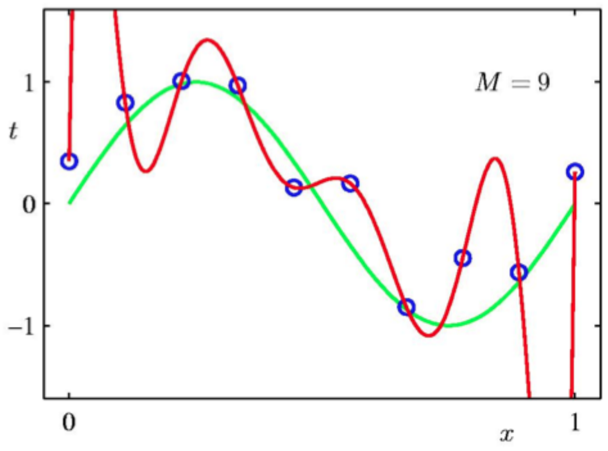
The red line shows the 1st order polynomial, while the green line is a sine wave with some noise, which is why the points are not located on it. It is easy to see that the polynomial fails to describe the points.

The second example is a much better fit. The model order is now 3.



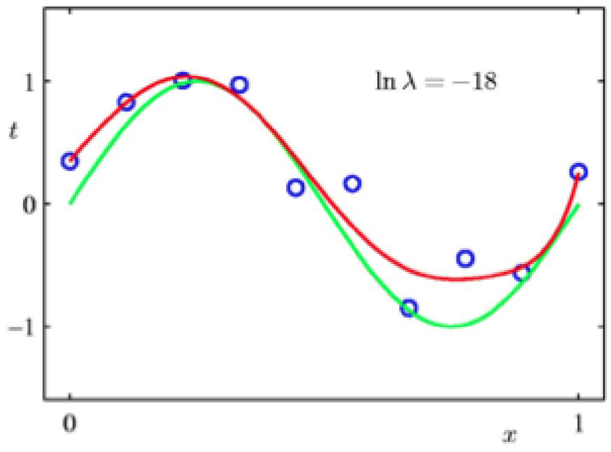
However increasing the order of the polynomial doesn't always make thing better, as shown in the next example where the order is 9.

[2]The images are taking from slides from the course 02582 - Computational Data Analysis



Now, instead of describing the underlying relationship between the points of data, the polynomial actually describes the random noise, which is not in our interest. An overfit model will generally not be very good as a predictive model, because it memorizes the known training data rather than learning to generalize it.

To avoid overfitting, one can use techniques to indicate when further training of the model does not give a better generalization. One of these methods is called *regularization*, which we won't go into details with, but what it does is to penalize the model for being very complex. With the addition of regularization to the model, the 9th order polynomial would look like the following:



3.4.1 Reducing the error

It can be hard to estimate the correct period lengths of a data set when using only one persons movement pattern. If a student for example is participating at a lecture every Thursday and the data describes his presence at that location it will be easy to estimate that the data is periodic with a frequency f. However if the person has missed some lectures the data will be noisy and it will be harder to estimate the periods in the data. We can use a method to reduce the noise of the data and in that way reduce the error e in our estimated model. If we measure movement patterns of groups instead of individual people we can reduce the noise in the data. The groups should contain people with similar location patterns, so at DTU for example students which are studying the same will more likely be placed in the same groups. When predicting a persons location the data from the group that the person is a part of will instead be used for the prediction. The correlation between different individuals' location pattern is used to define the groups. It's easy to calculate the correlation between persons by using the following method:

At a given time t we define a matrix X which have 2 dimensions. One dimension with students N and one dimension with all the locations L. The matrix holds information that describes all peoples location at time t. We can get the correlation between all the students by the following equation where C is the correlation matrix:

$X * X^T = C$

The matrix C will be an NxN matrix where every value in the matrix gives the correlation between two persons at the time t. Collecting data in T hours will result in T correlation matrices. By adding these T matrices to each other and normalize the data will give a correlation matrix which gives the correlation between every student when observing for T hours. The bigger time interval the more precise is the correlation matrix. The matlab function *corr(X)* can also be used to calculate the corrlation matrix.

3.5 Other approaches

The matrix C will be an NxN matrix where every value in the matrix gives the correlation between two persons at the time t. Collecting data in T hours will result in T correlation matrices. By adding these T matrices to each other and normalize the data will give a correlation matrix which gives the correlation between every student when observing for T hours. The bigger time interval the more precise is the correlation matrix.

4 Development

4.1 Android

As the client we have chosen to use Android. It is an open platform that is mainly developed by Google. It is based on a Linux kernel. It is for most cases used in smartphones. In our case the function of the Android device is to provide data to our webservice and also show results returned from the server. We chose this platform because it is open and lets you record the required data, compared to the iPhone that does not allow such recordings. It is a device that you will always have in your pocket so it is ideal to use it to track your location. By using the Android device to send location information with a timeinterval, we are able to get very precise location data for each user while they are at the DTU campus.

4.1.1 Location

To track location we needed a system that can determinate which building a user is currently in. We had the following goals for the solution:

- It should be able to determinate which building a user is currently in, so we wanted some accuracy.
- The system should be able to improve over time so it could determinate in which room the user was.
- Because of limited battery capacity on a mobile phone we wanted a system that did not drain the battery too fast.

To determinate a location you can use different technologies. Normally a smartphone uses 3 different technologies:

1. By look at the antenna towers located nearby and calculating the distance to each by looking at the signal strength for each tower.The advantage of this approach is the battery usage, as it is very low, but the precision is not good enough for what we need. You will only be able to determinate which area the user is currently in.
2. The most common approach is using GPS. All newer phones comes with a build in GPS that can determinate a location by communicating with the satellites. Communicating with satellites requires a lot of power, so this solution uses a lot of battery. The advantage of this approach is the precision. It can determinate the position very precisely, but only when you are outside. If you are in a building, it can often be very unprecise.
3. The last solution that is used in smartphones to determinate location is looking at Wi-Fi. A company called Skyhook Wireless has developed this technology. They have used the concept of Wardriving to search for wireless network, saving the location and MAC-address. They can then determinate a location by searching for wireless networks and checking if they are known in the database. Because WiFi hotspots usually are located inside buildings, this solution is ideal for what we want to accomplish.

We decided to use the idea behind last solution to create our own database of wireless networks on the Campus of DTU and then use that information to track the location of a user. We needed location information about where every WI-FI hotspot on campus. Luckily the Cisco Management System used by DTU contained that information. We combined that information with coordinates for every build from Google Maps and then we had our database.

The system can easily be improved in the future by improving the database to accomplish more precision and be able to determinate specific room instead of just building.

To retrieve information about the WI-FI hotspots that the device can currently reach, we use a singleton from the Android SDK called WifiManager. We can use this object to scan for Wi-Fi like this:

```
WifiManager wifiManager = (WifiManager) context.getSystemService(Context.WIFI_SERVICE);
wifiManager.startScan();
```

We will send this information to our webservice, that will handle everything.

4.1.2 Service

Normally an Android application will stop executing code after it has been brought to the background by the user. We wants to keep tracking a user location even though the user is using another app or the phone is in the pocket. In order to do that we created a Service. It is a part of the Android SDK. It lets you run a progress in the background even after the application has been brought to the background. We use this service to track the location and send MAC-addresses and signal-strengths for every wireless-network it can see at the moment. We choose that our service should be called every 10 minutes. Every time it runs, it will make an observation and send it to our webservice.

4.1.3 Communication to server

When we have an observation, we send it to the server. We do this through SOAP, which is described in section 4.2.3. In order to establish the connection, we use a framework called ksoap2-android. We created a class called Webservice to handle all the communication with the server. I works very simple. You just specify which method on the server you want to call and add the parameters that the method takes. Then you call the method and get a response back.

4.2 Webservice

To store all the data from the client and to do the prediction calculations we use a web service.

4.2.1 Server

The original plan was to use a server located at DTU. The idea was that all the groups could use the same server. The plan then was to setup a framework for all the groups to use, and in that way make it easier for the projects to cooperation. Because of delay in getting a server for all the projects, we decided to set up our own. We rented a Amazon EC2 server. This allowed us to create an virtual machine. We choose Amazon cloud server because it is very good for scaleability. You can at any time upgrade the CPU or add more memory. Another advantage was that you can choose only to pay for the CPU time and data usage. Because we only were going to send a small amount of date in the beginning, this was the perfect solution.

With the Amazon EC2 we had a server that we could do whatever we wanted with. Based on our experience, it was later decided that all the projects should run on an Amazon EC2.

4.2.2 RESTful

Our first approach was to use a a RESTful architecture. A REST server in based on a simple concept. The client makes a HTTP request to the server. As a response the server will usually return XML or JSON. Because the protocol used was HTTP with could perform GET, POST, DELETE etc. to the server. We used a framework called Play. It is a java based web framework. We connected play to a MYSQL database so we could easily store our date in tables.

We defined and implemented the following protocol for the Play server:

Request parameters

GET /location/{user_id}/newest

Parameter	Type	Description	Example
user_id	String	Client identification used for authentication and authorization.	423rsdfs43fsdf

Response

Return HTTP/400 if parameters are missing or request is bad. Will return HTTP/401 if the user is does not have any recordings. If the request is valid the response will be a HTTP/200 OK. The response will be in JSON.

Response parameters

Parameter	Type	Description	Example
lat	String	The latitude of the last newest position	55.788109
lon	String	The longitude of the last newest position	12.523642
record_date	String	Timestamp when location was recorded	2011-05-20 10:34:12

Request parameters

GET /location/{user_id}/{recordings}

Parameter	Type	Description	Example
user_id	String	Client identification used for authentication and authorization.	423rsdfs43fsdf
recordings	Integer	The number of the latest recordings wanted for a user	10

Response

Return HTTP/400 if parameters are missing or request is bad. Will return HTTP/401 if the user is does not have any recordings. If the request is valid the response will be a

HTTP/200 OK. The response will be in JSON.

Response parameters

Parameter	Type	Description	Example
list	Complex	The list of location items for the user	N/A

Location item

Parameter	Type	Description	Example
lat	String	The latitude of the position	55.788109
lon	String	The longitude of the position	12.523642
record_date	String	Timestamp when location was recorded	2011-05-20 10:34:12

Request parameters

POST /location/{user_id}/lat/{lat}/lon/{lon}/date/{date}

Parameter	Type	Description	Example
user_id	String	Client identification	423rsdfs43fsdf
lat	String	The latitude of the position	55.788109
lon	String	The longitude of the position	12.523642
record_date	String	Timestamp when location was recorded	2011-05-20 10:34:12

Response

Return HTTP/400 if parameters are missing or request is bad. If the request is valid the response will be a HTTP/200 OK. The response will be in JSON. **Response parameters**

Parameter	Type	Description	Example
Status	String	The status of the POST	OK

As shown above we had 3 different calls to the webservice. You could get the newest location of a user from a userid. You could get a specific number of newest locations from a userid and you could post a location for a user.

Because is was later decided from a higher place to use a different framework for all the projects, we had to leave this server approach.

4.2.3 SOAP

As mentioned it was decided that we should use a common framework on the server. The decision ended up being a Amazon EC2 server with Apache HTTP Server, Apache Tomcat, Apache Ant, Axis2 and a MYSQL database. You can read more about the MYSQL database in section 4.3. As it would take some time before the server for everyone would be up and running, we decided to install everything on our own EC2 server and we installed a localhost on our computers as well, so I was easier to test while we were developing the server.

Our server is running the following frameworks:

Apache Tomcat

We use Tomcat as the servlet on the server. It makes it easy to make HTTP requests to the server. By using Tomcat we can use the request-response model.

Axis2

Axis2 is our core engine. It works together with SOAP and we use it to send messages from our client to the server. Our server support 5 different methods.

- **login(String username, String password)**
  This methods is used to register a user in the database. If the user is not yet created, it will be created. It will create a session for the user. It does that by that by generating a random string called the session hash. This string will be saved in the database associated with the id of the user. The client will as a response receive the session hash. In this case the client is our Android application. For every other method that the client want to call on the webservice it have to provide this session hash. In that way it is easy to check whenever a user have access to a specific method.

- **logout(String sessionHash)**
  This method is used to logout a user. It will remove the session hash from the table with sessions.

- **observe(String sessionHash, String jsonString)**
  This method makes it possible to register an observation. The client will have to pass in the session hash and a string in JSON format that holds information about the observation. This is an example of such a string:

  ```
  {"observedSignalStrengths":[-71.0,-71.0,-72.0,-83.0,-84.0,-85.0,-88.0,-89.0,
  -89.0,-89.0],"macAddress":"F8:DB:7F:43:99:29","observedMacAddresses":
  ["00:1c:0e:42:77:3","00:1c:0e:42:77:3","00:1c:0e:42:77:3","00:1c:0e:42:84:2",
  "00:1c:0e:42:84:2","00:1c:0e:42:84:2","00:1c:0e:42:80:1","00:1c:0e:42:80:1",
  "00:02:72:8b:60:0","00:1c:0e:42:80:1"],"deviceType":"WIFI"}
  ```

  The information provided is: observedSignalStrengths: An array of all the signal strengths

  observedMacAddresses: An array of all the MAC-addresses. (Indexes must match signal strength)

  macAddress: The MAC-address of the client. In our case the Android device.

  deviceType: The type of the observed MAC-addresses. In our case it will always be "WIFI", but the server also supports observations from bluetooth. This is supported in order to integrate our database with the one of the other groups from the collaboration.

  The methods will return a JSON string indicating if the observation was successfully inserted in the database. For example:

  `{"error":"", "statusCode":1}` If the observation was successfully inserted.
  `{"error":"Invalid session", "statusCode":0}` If the provided sessions hash was not valid.

  On the server and on the client we use a library to easily parse the JSON called Gson. It a library developed by Google and lets you create java object from JSON strings. For example we can create a Double object with the value of the latitude from a JSON string like this:

```
GsonBuilder gsonb = new GsonBuilder();
Gson gson = gsonb.create();
Double latitude= gson.fromJson(jsonString, Double.class);
```

- **getNewestUserPosition(String sessionHash, String user)**
  By providing the session hash and the username of a user, this method will return a JSON string with the position of the user and the timestamp for it. It will look the position up in the database. We will talk more about that in the Hibernate section.
- **getAllUsersNewestPosition(String sessionHash)**
  Will return the newest position for all the users.

Hibernate

To generate the tables in our database, populate it and later retrieve information, we are using a framework called Hibernate. It is a framework that allows us to create a mapping between the tables in the database and objects in Java. We use the Hibernate Query Language (HQL) to populate and retrieve objects from the database. HQL is a simple readable format that looks a lot like SQL. For example if we want to get a device object from a MAC-address, we can look it up in the database by writing:

```
Device d = (Device)session.createQuery("from Device d where d.macAddress =
'" + macAddress + "'").uniqueResult();
```

This will return a device object with the given MAC-address. We can then access all of the fields for that object.

Hibernate also support executing SQL directly in the database. We use this when we want to select all the newest positions of all the users. We are using the following SQL query.

```
SELECT id, latitude, longitude, date, name
FROM = (
SELECT location.id, location.latitude, location.longitude, location.date, user.↩
    name
FROM = (
user
INNER JOIN user_device ON user.id = user_device.user_id
INNER JOIN device ON user_device.devices_id = device.id
INNER JOIN device_location ON device.id = device_location.device_id
INNER JOIN location ON device_location.locations_id = location.id
)
GROUP BY location.date DESC
) AS T
GROUP by name
```

What we do is simply joining all the tables and group the result by location date and afterwards by name. This will give us the newest location for every user.

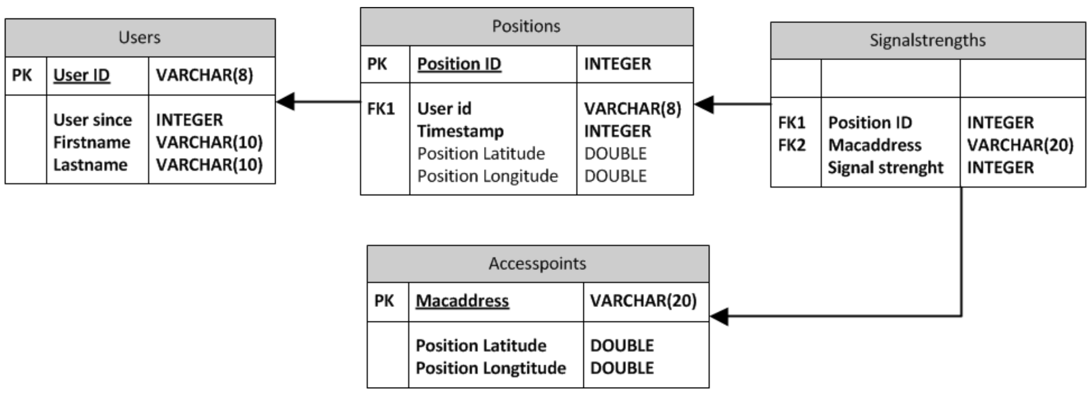
Apache Ant

We installed Apache Ant on the server. This lets us easily build the project using the Ant command. Ant uses a file called build.xml. In this file we have specified everything that we want to happen when running the ant command. In our build.xml. It will start up

Tomcat and build the database using Hibernate. It will also install our java application on the server.

4.3 Database

At one of the meetings with the other groups of the CampusNet app we agreed to setup a server with a database that we all could access and use. At the time we already created a database diagram that fitted our needs. It is illustrated below:



Users		
PK	User ID	VARCHAR(8)
	User since Firstname Lastname	INTEGER VARCHAR(10) VARCHAR(10)

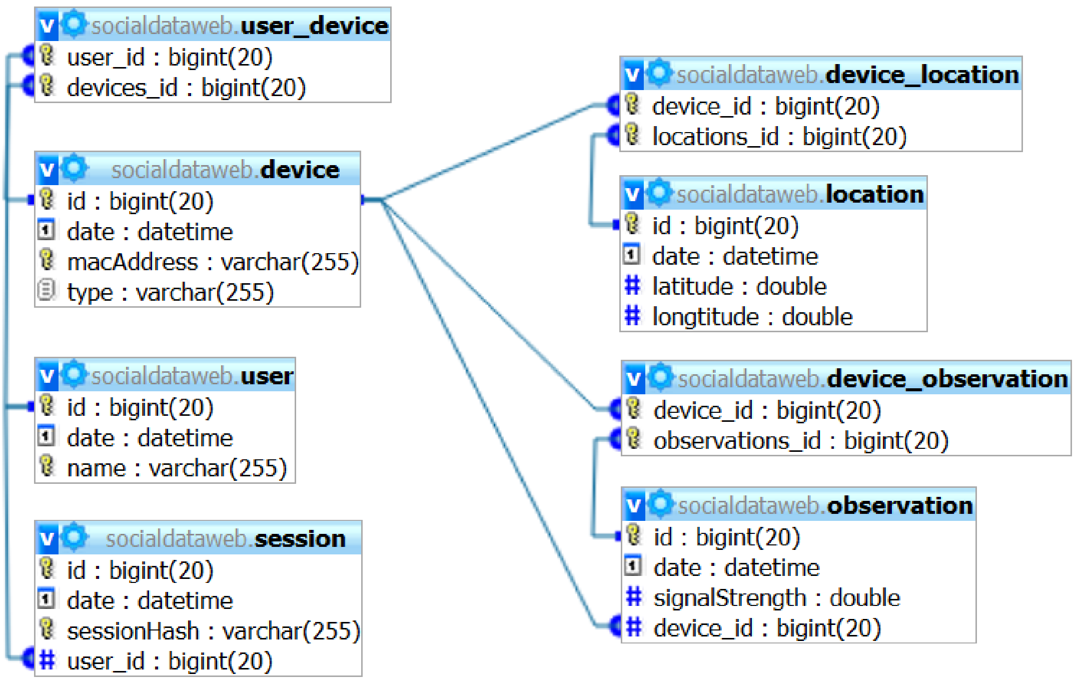
Positions		
PK	Position ID	INTEGER
FK1	User id Timestamp Position Latitude Position Longitude	VARCHAR(8) INTEGER DOUBLE DOUBLE

Signalstrengths		
FK1 FK2	Position ID Macaddress Signal strenght	INTEGER VARCHAR(20) INTEGER

Accesspoints		
PK	Macaddress	VARCHAR(20)
	Position Latitude Position Longtitude	DOUBLE DOUBLE

The diagram describes the different tables of the database and the data types of the values that needs to be stored. The data structure is separated in four tables so we store the data the most logic way and use as less memory space as possible. We could also have written all the data in one table but that wouldn't be intuitive and we will end up with a lot of rows with empty columns. The tables relate to each other through different id's. A user has for example has several positions stored with a different timestamps. The positions relate to the user by the users id which is the users study number and we can easy get a users position by his name if we join the to tables on the user id.
Referring to the two flow chart diagrams in section "Processes" the database stores the incoming access points signal strengths in the table "Signalstrengths" and the server calculates the user position based on the signal strengths data and stores it in the "Position" table. The Accesspoints table is already filled with our known access points MAC-addresses and their positions and the Users table stores the user when the person logs into the Campusnet app the first time.

Because we are collaborating with several groups we needed to find a database structure that would fit everybody's needs. For this reason we came up with this model:



For looking up the coordinates of a specific Mac-address we added another table macaddress_location. In order to get the needed data we combined information about the building of each wireless-hotspot provided by Cisco Management System at DTU with locations of the each building from Google. We filled our database with this information and ended up with a table consisting of Mac-address, latitude and longitude.

A typical flow would be like this. When a user is trying to login for the first time he will be registered in the user table and he will get a session in the session table. If he at some point choose to logout his session will be deleted. When a user observes an observation for the first time with a device, the MAC-address of the device will be saved in the device table. The MAC-address of all the WIFI-hotspots that the user observed will also be saved as a device. Further more every device will be associated with an observation that holds information about the signal strength. If the MAC-address of the device is in the table of known locations for MAC-addresses, the location of the device will automatically be added to the location table.

4.4 Matlab

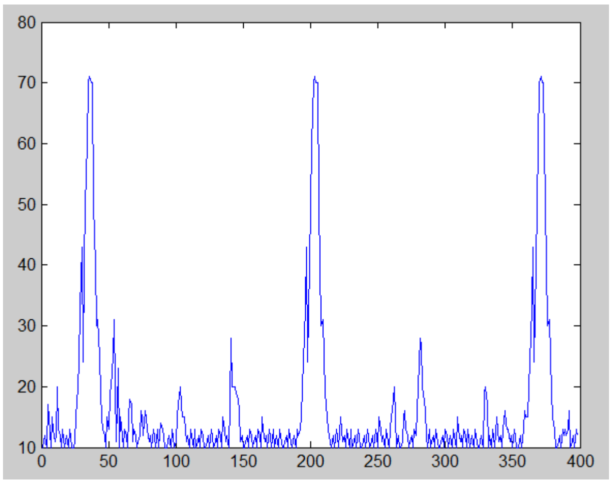
The prediction algorithm won't be implemented as a finished module but we will code parts of the prediction algorithm in Matlab and call it a prototype. The Matlab code is in the appendix.

In this section we will describe how the different parts has been implemented and later test these parts with some test data. The correlation part where the groups is getting defined wont be implemented but the method to do this is described in the section *reducing error*.

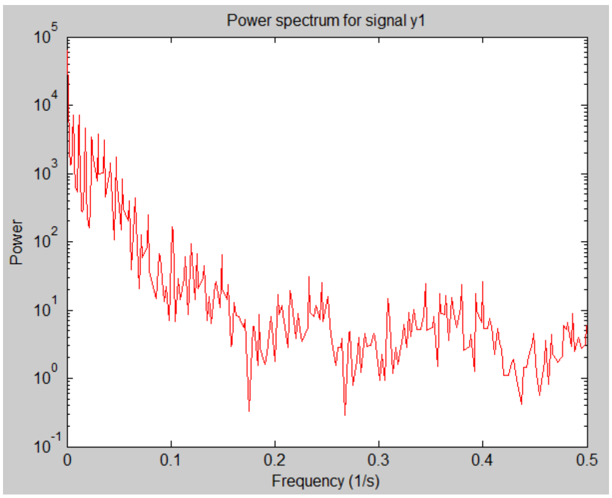
The part which is going to be implemented is a module that can take in some data about the location of a group and fit a model that can predict future locations of the people in the group.

The function *ar(y ,sb, 'yw')* will be used to estimate the model. The 'yw' string means that the Yule Walker equations will be used to calculate the coefficients of the AR model. The sb variable gives the number of steps the algorithm should "look back" to predict the next value. The size of the variable depends on the length of our data.

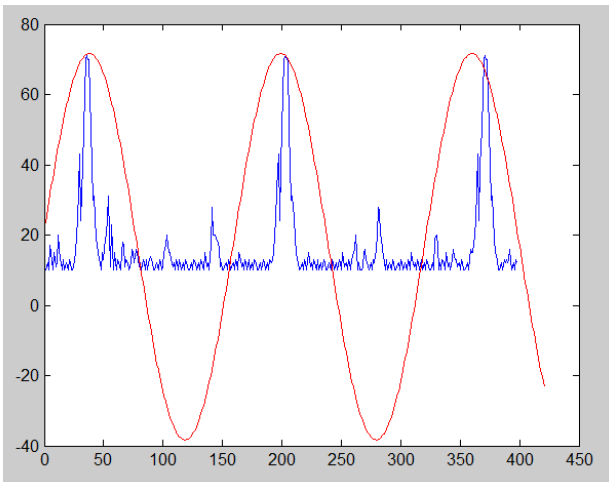
Lets consider the data for one access point in this graph:



The graph gives the number of people located at the specific access point at time t. Its easy to see that the data is periodic and the method used to find this periodic frequency is the matlab function *ffplot()* that gives the power spectrum of the data.



Its also possible to use a fourier transformation instead of the AR model to find this spectrum. From the power spectrum the matlab algorithm picks out the highest maximums to estimate a sin model that fits the data.

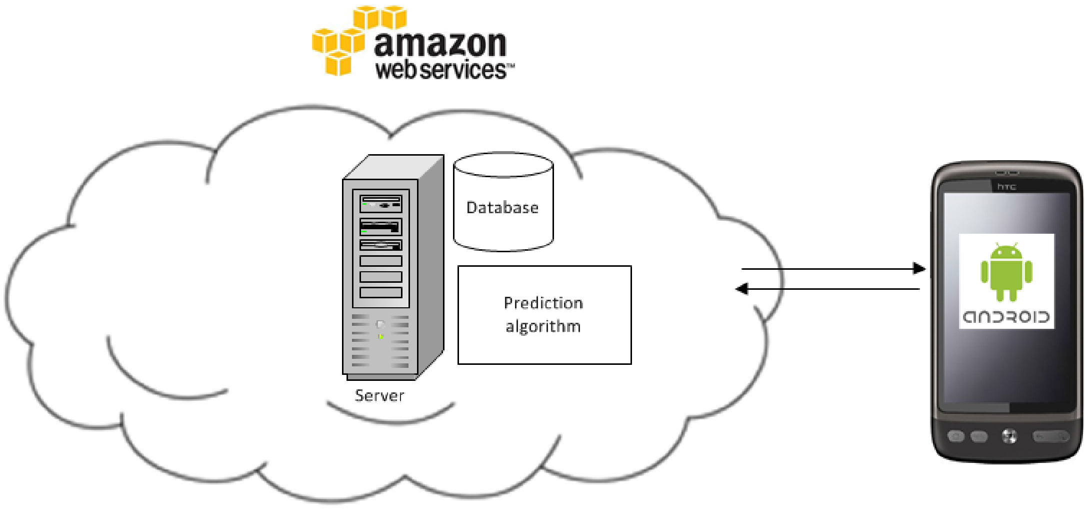


The mean of the sine-curve is equal to the mean of the data and the amplitude gets calculated from the maximum values. The sin curve can be used to predict with some probability if the group is at the same access point at time t where t is a future timestamp. We can predict at which location the group is located when using data from all the different accespoints. From this data we can pick out the accespoint which has the highest probability of where the group is located at time t and use this location as the prediction.

The matlab code for this algorithm haven't been deployed on the web-server. However this can be done by creating an matlab executable on the web-server that can take some input and provide the predicted data as output. It's also possible to convert the code to java code with the matlab module *deploytool*.

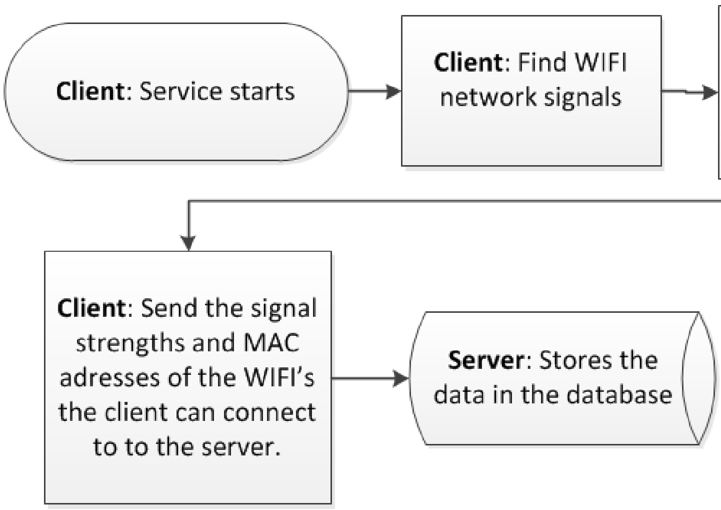
4.5 Architecture

The architecture of the whole system is illustrated below:



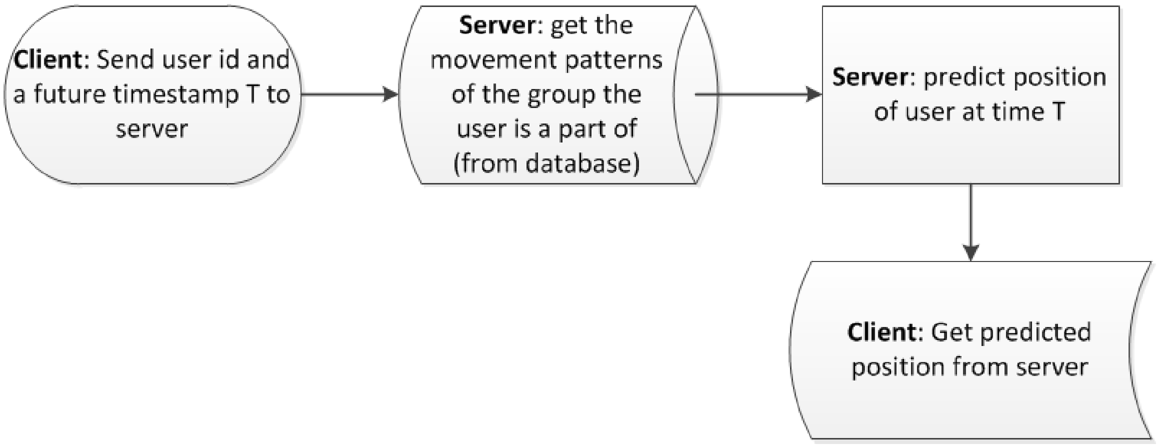
The system contains one server in the cloud which communicates with several Android clients via HTTP-requests. It is wise to think of best way to place the different algorithms so they run fast and are easy to maintain and extend. For this reason we let the server do the prediction because it's more powerful and easier to modify when its located in one place. Its also better to let the android phones do as few calculations as possible because it have fewer resources and runs on battery. There is many different technologies surrounding the project and it's important that these technologies is well integrated and its important to use a strategy that describe how the different frameworks should communicate with each other. Every part of the project is programmed in Java and that simplifies the data transfers. It is for example easy to write an algorithm in Matlab convert it to java code and integrate the code with the axis 2 framework which is running on the server.

In order to post an observation to the server from the client, we use the following flow:


Client: Service starts
Client: Find WIFI network signals
Client: Measure signal strength for all WIFI networks
Client: Send the signal strengths and MAC adresses of the WIFI's the client can connect to to the server.
Server: Stores the data in the database

First the service that will fire every 10 minutes starts and collect the required date. It sends the data to the server where it will be stored in the database.

Then plan for how a prediction should work is illustrated below.

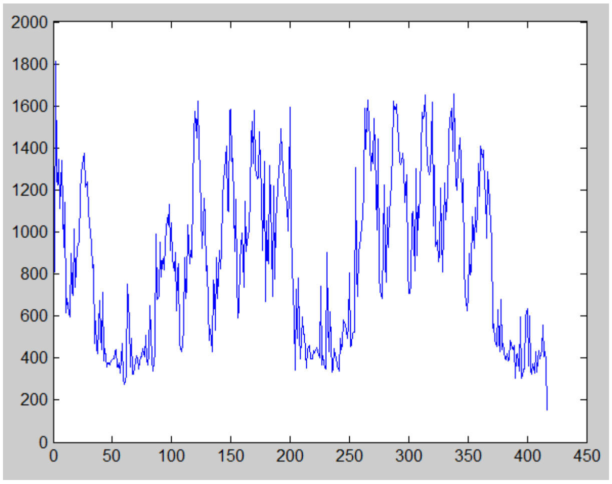

Client: Send user id and a future timestamp T to server
Server: get the movement patterns of the group the user is a part of (from database)
Server: predict position of user at time T
Client: Get predicted position from server

The user sends the id and a timestamp for when want to get a predicted position. The server will lookup the movement patterns for the user with the provided userid and then use the prediction algorithm to calculate the position. It will then return that position to the server.

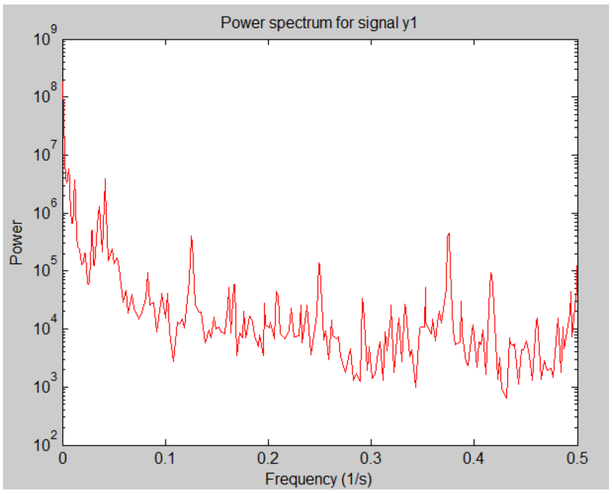
5 Test

5.1 Matlab module

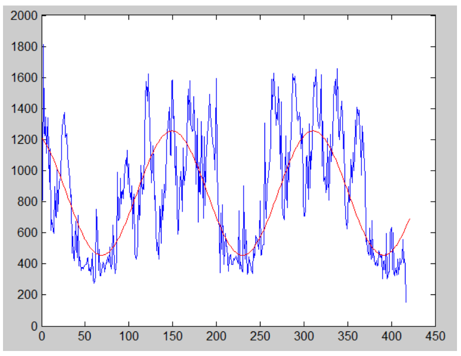
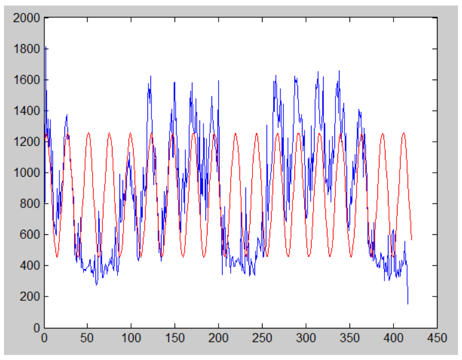
In order to test the matlab module we will provide it with some data and let it calculate a model that fits the data. The module needs to do the most by its own but we need to adjust the module a little bit so it includes the number of different frequencies in the module which fits the data best.
The testdata is provided by the PHD student Toke Jansen Hansen and it describes a groups position near one accesspoint at different timestamps. Lets consider the dataset and imagine that it describes the position of a study group at a access point which is located in their classroom.



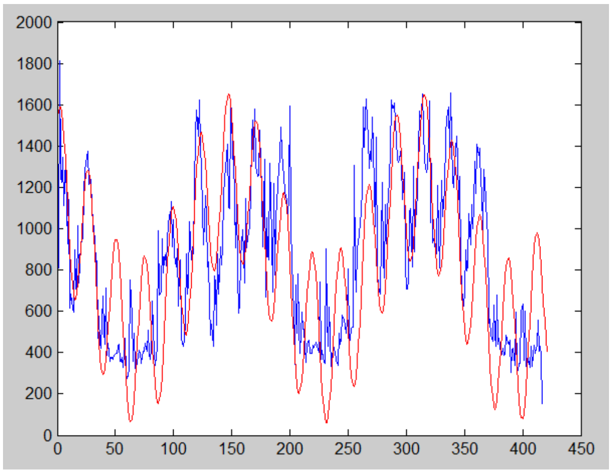
The AR module provides the power spectrum of the data.



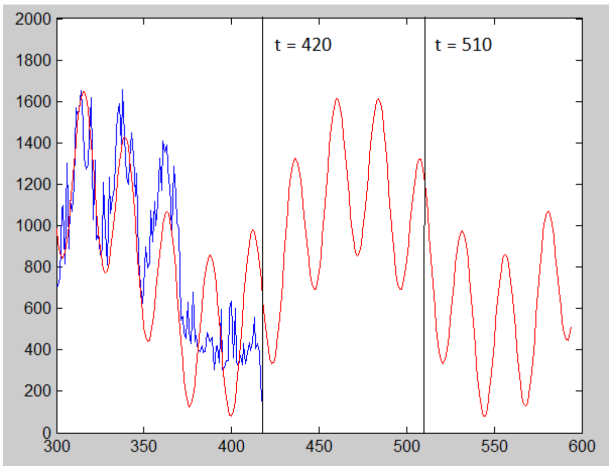
By picking out two of the highest peaks we get two different sine-curves. One with a period-length of a day and one with a period-length of a week.

It's not strange that the data is acting this way because it's obliviously more likely that the students are at school in weekdays and in the day hours. We can combine the two sin curves by adding them to each other and get a model that fits the data well.

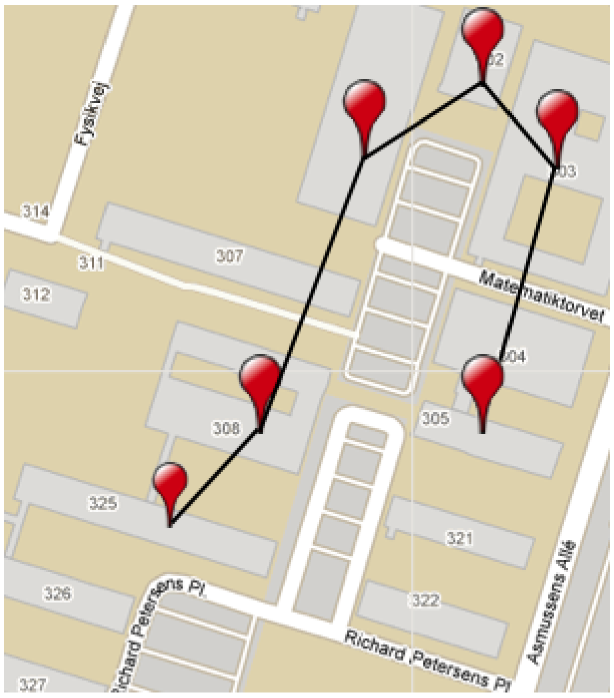


This model can as described in the previous sections be used to predict if the group is located in the classroom at a specific time. The graph below shows a prediction done at time t = 520. The prediction at time 510 shows that there is a high possibility that the group is located in the classroom.



5.2 Webservice test

Below is a test we made after setting up the webservice. The figure shows actual data plotted on a map of DTU. The test is made by moving one device through different areas, having it send it's position with small time intervals. The test run tracked the position, as shown in the figure, which is a visual representation of the data.



6 Further development

As our project still is very much in the prototype stage, it holds a lot of room for improvements. Some of these improvements, and ways of making them, are listen below.

6.1 Location determination

The way the location determination works now is by look at WIFI's. However this does not give a very clear indication of where the actual device is located, as the WIFI ranges can be very large. In addition to using the WIFI locations, one could choose to include the GPS location, and if the WIFI signals are too weak then the application switches over to using the GPS.
One of the other projects in our correlation with the DTU CampusNet was concentrated around finding people's location inside buildings to a room level of precision. They've done this by measuring WIFI signal strengths in each room, so that each room has assigned a set of signal strengths which are unique for that room. Incorporating their work into ours would give us a much more detailed database, which would allow us to track the users position much more accurately.

6.2 Predictions

As it was mentioned earlier, there is no definite answer to how you make the best prediction algorithm. Once a lot of data has been collected, it could be interesting to implement other types of prediction algorithms and hold these up against each other, to see if one is more effective than others.

6.3 User experience

Because this project does not give much back to the user, a fun feature to implement could be a game aspect in which the user competes against the prediction algorithm, so that the user gives a guess of where himself or a friend will be in the future while the program does the same, and eventually the user can see if he knows his friends better than the program.
The user experience could also be improved by developing the social aspect of the application, to let users add each other as friends, rather than just being able to track everyone who uses the application, which would not work in reality and probably violate some people's privacy.

7 Discussion & Conclusion

With our project we have taken some small steps towards an answer, when it comes to determining the predictability of people, and even though our project is still in the prototype phase, the gathered data is stored on a database, and can therefore always be used later on. The introduction of mobile- and especially smartphones has provided solutions to questions that couldn't be answered before. One of them is the problem that we've worked with, namely prediction of people's positions.
The work done with the prediction algorithm gives an understanding of the general principles when it comes to prediction of time series, but if it ever was to be used practically it would require some tweaking and adjustment. It could also be interesting to implement some other types of prediction methods so that these can be compared in order to find the best solution. By analyzing test data, we have found that the movement patterns have some sort of periodicity, that can be used in a prediction.
While developing the server, we had to try two different approaches, and in some way ended up doing the same work twice, so in that regard we could have saved some time, but the set up of two different kinds of webservices has given us some insight in the advantages and disadvantages of both.
The collaboration we've been a part of is something we haven't tried before, and it has been interesting because the work done by some of the other involved groups is highly relevant to our project and could help us improve our work if it was merged into one application.
While it is very interesting to find out whether or not people are predictable, it does bring up some ethical problems concerning peoples privacy, and if the application was to be used by students at DTU or anywhere else, you would have to take this into consideration.

8 Referencer

- http://developer.android.com/reference/packages.html
- http://axis.apache.org/axis2/java/core/docs/
- http://en.wikipedia.org/wiki/Apache_Tomcat
- http://www.hibernate.org/docs
- http://en.wikipedia.org/wiki/Hibernate_(Java)
- http://en.wikipedia.org/wiki/Hibernate_(Java)
- http://www.stat.ufl.edu/ berg/sta4853/files/sta4853-10print.pdf
- http://en.wikipedia.org/wiki/Autoregressive_model#Yule-Walker_equations
- http://en.wikipedia.org/wiki/Overfitting

Appendix

```
function [ ] = powerspecDataARFunction( data )

%POWERSPECDATAARFUNCTION Summary of this function goes here
%   Detailed explanation goes here

data = double(data);
maxvalue = max(data,[],1);
mean = Mean(data)
periodStartPos = 0;
x = 1:size(data, 1);

y = iddata(data);
arModel = ar(y ,216, 'yw');

%% Periode day estimation:
[amp,phas,w,sdamp,sdphas] = bodeaux(0,arModel);
w=w/2/pi;

figure(1)
plot(data)
%Plot arModel
figure(2)
ffplot(arModel, 'r')

figure(3)
u = iddata([],idinput(900));
yh = sim(arModel,u)
plot(data, 'b')
hold on
plot(yh, 'r')
hold off


%% Periode week estimation:

peak = 0;
index = 0;
ampArray = [];
for i = 1:size(amp,3)

    value = amp(:,:,i);
    ampArray = [ampArray value];
    if amp(:,:,i) > peak

        if i>2 && amp(:,:,i-2) < value && amp(:,:,i+2) < value
            peak = amp(:,:,i);
            index = i;
        end
    end
end

amp;
phas;
w;
sdamp;
sdphas;
index
f = w(index)


figure(4)
plot(data, 'b')
hold on
%periode:
T = 1/f
T2 = yh(3);
```

```
x = pi : 135*pi;
A = 55
K = (2*pi)/T;
sinWeekFit = A*sin(K*(x+periodStartPos))+mean';
plot(sinWeekFit, 'r')
hold off




end
```

```
%Test of algorithm
close all
clear all
clc

tmp = load('cover_posts_per_hour.mat', 'posts_per_hour');
matrix = tmp.('posts_per_hour');
data = matrix(2,:)';
data = double(data);
data = data(16:432);
mean = Mean(data)
x = 1:432-16;
y = iddata(data);
arModel = ar(y ,216, 'yw');

%% Periode day estimation:
[amp,phas,w,sdamp,sdphas] = bodeaux(0,arModel);
w=w/2/pi;

peak = 0;
index = 0;
ampArray = [];
for i = 1:size(amp,3)

    value = amp(:,:,i);
    ampArray = [ampArray value];
    if amp(:,:,i) > peak

        if i>51 && amp(:,:,i-2) < value && amp(:,:,i+2) < value
            peak = amp(:,:,i);
            index = i;
        end
    end
end

f = w(index);

%Plot arModel
figure(2)
ffplot(arModel, 'r')

figure(3)
u = iddata([],idinput(900));
yh = sim(arModel,u)
plot(data, 'b')
hold on
plot(yh, 'r')
hold off


figure(4)
plot(data, 'b')
hold on
%periode:
T = 1/f
```

```
T2 = yh(3);


x = pi : 190*pi;
A = 400;
periodStartPos = 1;
K = (2*pi)/T;
sinDayFit = A*sin(K*(x+periodStartPos))+mean';
plot(sinDayFit, 'r')
hold off

%% Periode week estimation:

peak = 0;
index = 0;
ampArray = [];
for i = 1:size(amp,3)

    value = amp(:,:,i);
    ampArray = [ampArray value];
    if amp(:,:,i) > peak

        if i>2 && amp(:,:,i-2) < value && amp(:,:,i+2) < value
            peak = amp(:,:,i);
            index = i;
        end
    end
end

f = w(index)

figure(5)
plot(data, 'b')
hold on
%periode:
T = 1/f
T2 = yh(3);

x = pi : 190*pi;
A = 400;
periodStartPos = 50;
K = (2*pi)/T;
sinWeekFit = A*sin(K*(x+periodStartPos))+mean';
plot(sinWeekFit, 'r')
hold off

%% Combining the two

figure(6)
plot(data, 'b')
hold on
%periode:
T = 1/f
T2 = yh(3);
sinCombinedFit = sinDayFit + sinWeekFit - mean;
plot(sinCombinedFit, 'r')
hold off
```