

## **Applying Position Prediction as a Means for Performance-tuning in Location-aware Platforms**

Andreas Post, Peter Langendörfer and Rolf Kraemer

IHP, Frankfurt / Oder, Germany, e-mail: (post, langendoerfer, kraemer)@ihp-microelectronics.com

*Abstract - With the constantly increasing number of mobile devices performance becomes more and more essential for location aware platforms. Position prediction can help to improve the overall performance of such location-aware platform, e.g. by reducing latencies during handovers. But this goal can only be achieved if the prediction mechanism provides a high accuracy and if it does not consume relevant computing resources. Here we are discussing a simple position prediction approach that meets both requirements. Depending on the parameters for position update and position prediction its accuracy is better than 85 per cent. In addition a single prediction is calculated in less than 0.02 millisecond on a state of the art processor.*

### **1 Introduction**

Location-aware systems and services have attracted a lot of attention during the recent years and a lot of research was done in this area. The functional issues, e.g. how to relate information to the current position of a user are more or less solved. We are concentrating on non-functional topics like performance tuning. In this paper we introduce a simple and efficient a position prediction unit, and discuss its benefits for location aware services and platforms. Due to our focus on indoor scenarios and hot spots we are mainly interested in prediction periods of few seconds.

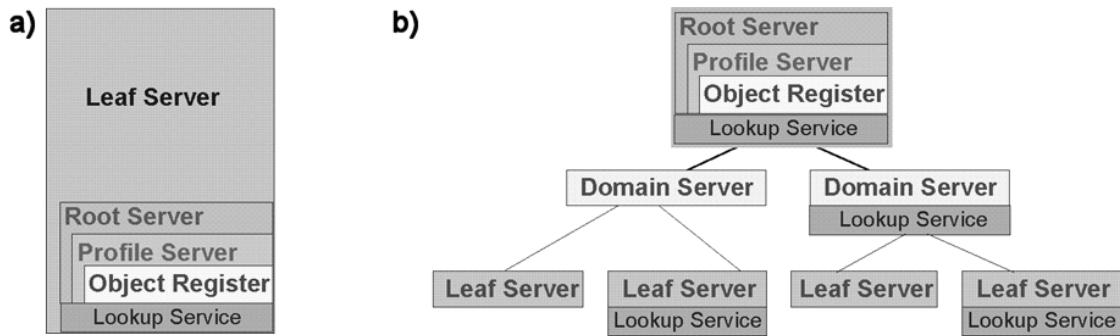
Our simulation results show that our position prediction approach delivers very good results. With a position update per second and a prediction of the next position i.e. we look in the future for one second, more than eighty-five per cent of the predictions are correct. Please note that a single position prediction needs less than 0.08 milliseconds on a Pentium 3 with 450 Mhz. Despite we used our own location-aware platform, called PLASMA, to evaluate our approach it can be used in any other system.

We start with a short introduction of PLASMA, then we explain the position prediction mechanism applied. Thereafter we discuss the realization of our simulation set-up and how it is integrated in the platform. We will briefly discuss our simulation results. This paper concludes with an outlook on further research steps.

### **2 PLASMA**

The PLASMA infrastructure consists of Root Servers, Domain Servers, Profile Servers, Leaf Servers and Object registers [1]. The major design goals are adaptability, world wide

scalability and openness. PLASMA allows to configure e.g. which of the above mentioned servers are clustered together on a single machine as well as on which machine they are installed. Thus, it is possible to run all servers on a single machine (see figure 1a) in order to provide PLASMA functionality in a small hot spot such as a shopping mall or train station. If a large area has to be covered PLASMA servers may be structured as a tree. This allows to use e.g. a single object register for two or more physically separated hot spots (see figure 1b).



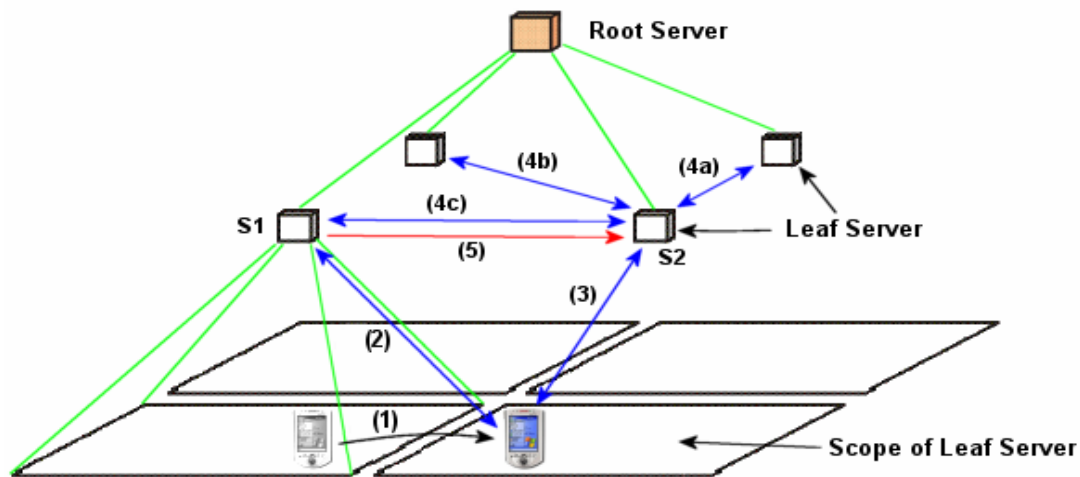
**Figure 1: PLASMA deployment scenarios: (a) single server scenario; (b) hierarchical structure**

In order to provide useful location aware applications it is necessary that the current position of a subscriber can be queried by the application. This means a suitable positioning system has to be part of the infrastructure and it must be able to submit its information to PLASMA. This information has to be processed in a suitable manner. In addition potential users require the possibility to define areas of interest which we call aura as well as means to define events such as “*notify me when a color printer is less than 5m away*”. In order to support large scale deployments a handover mechanism between is needed. The platform consists of the following components:

- database engine
- event engine
- auras & objects engine
- communication engine
- lookup engine
- profile engine
- hand-over engine
- sighting proxy

The sighting proxy allows the integration of several positioning systems into the infrastructure. Currently GPS and infra red beacons are supported. Each position information contains the positioning system, a timestamp and for sure the current position of the client. This information is stored in the database engine and can be used for position prediction. If several leaf servers are used in a PLASMA deployment the hand-over engine allows to transfer data such as the profile of the mobile clients from one leaf server to another. Currently this is done using a pull strategy (see figure 2). As soon as a mobile has moved from the scope of leaf server S1 into the scope of S2 (1), S1 informs the mobile that it has to

connect to a new leaf server (2). After the connection set-up with the mobile (3), S2 asks all its neighbors for related data (4a-c). This information is sent in (5). This platform internal handover takes 100 milliseconds in total. S1 needs about 30 milliseconds to gather and marshal the data, S2 needs 30 milliseconds for unmarshalling and distribution of the data to the responsible platform modules. The remaining time is communication overhead. At a first glance, the time needed for internal handover does not look too bad. Currently a leaf server can handle up to 1000 events per second. Thus, only three handovers per second reduce the



**Figure 2: Handover with pulling.** A mobile enters the aura of another leaf server (1). S1 informs the mobile that it is now in the scope of a different leaf server (2), After the connection set-up (3) the S2 asks its neighbours if any of them already knows the mobile 3a-c, data of the mobile is send from S1 in (4).

performance about ten per cent. Even with the assumption of a medium mobility it might happen that several handover operations have to be done simultaneously. Thus, the overall performance will decrease drastically. The idea here is to use the position prediction to improve the overall performance. With position prediction, hand-over operations can be done:

1. in advance, when it is detected that a mobile might leave the region of its current leaf server
2. in a period of time with low event processing load.

### 3 Related Work

There is already some work published in the field of position prediction. Relative popular are methods that use movement profiles called home-office-home pattern [2]. In [3] it is proposed to gather profiling data over a certain period of time, normally several days or even longer. The used method is called Regular Path Recognition. It is based on the assumption that user behavior is very regular, e.g. that a user takes every day the same way to work. Out of this it is possible to create patterns with end points like “home”, “work”, “sports” or something else that are connected through paths. Along these paths cell patterns and cell combinations are

saved to create user profiles. A counter is also saved that indicates how often a cell combination occurs. These profiles are used to predict the movement of the user.

In [4] neural networks are used to predict positions, which also requires a training phase. In the first phase the user movement was registered six times a day over four weeks. The output of the neural network is a list of base stations and the probability that the user is in the range of a certain base station. The system sends its request first to the base station with the highest probability. If the user doesn't react, the system sends an enquiry to the base stations with the next lower probability and so on until the user is reached.

In systems we have in mind are hot spots where people will be only for short period of time i.e. at most several hours. In these environments it is very hard to recognize patterns. This is due to the limited time and the fact that the user is in an unknown environment, so she might be interested in very thing, and changing her behavior every now and then. The situation may change when a certain user becomes a regular guest of a hot spot, but then profiling user movements may be prohibited for privacy reasons. These are the reasons why we investigated whether position prediction can be done on the basis of the last two or three positions of the user. In order to decide whether or not the profile of a user has to be forwarded it is only necessary to predict the position of the user one, two or three seconds in advance. Despite we are aware of the fact that the type of the positioning system has some influence on the accuracy of the prediction, we assume for the rest of this paper that the positioning information is perfect.

## 4 Position Prediction

### 4.1 Prediction Algorithm

The first necessary step for predicting a new position is to determine the path of the user. Due to the fact that we are only interested in a short term prediction (one to three seconds) we decided to use only the last two positions P1 and P2. Here P1 is the current position and P2 is the last position before P1 was reached. Using this information a straight line equation can be constructed where P1 represents a point vector and the difference (P2 – P1) represents a direction vector. Using the timestamps provided by the positioning system the current velocity of the user can be calculated. Our basic assumption is that the velocity will remain constant for the next few seconds. Thus, we use the following equation to calculate the next position:

$$\overline{P} = \overline{P_1} + (v * t) * \overline{P_2 - P_1}$$

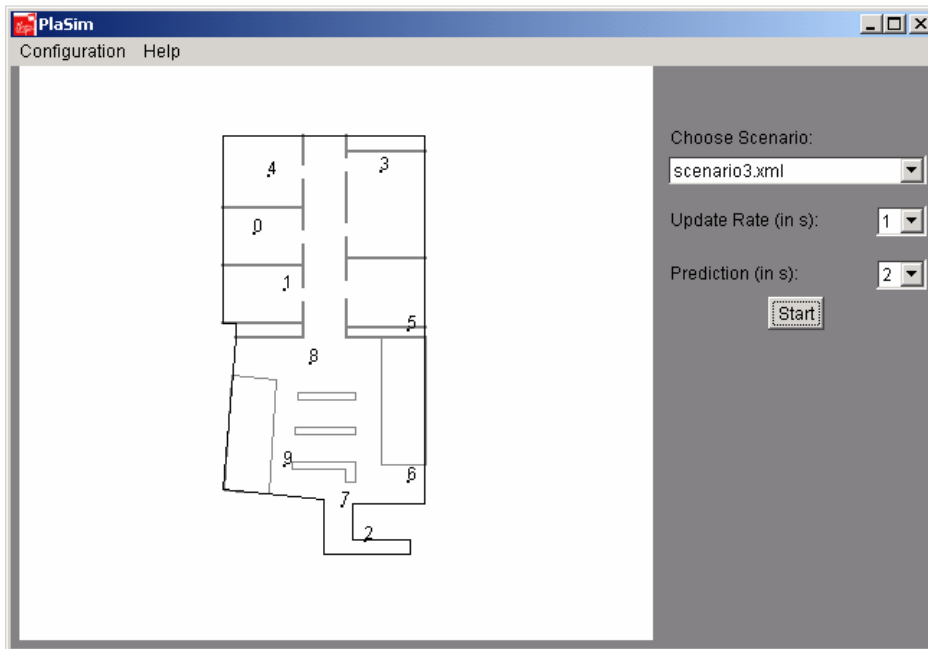
where  $v$  is the velocity and  $t$  is the time in milliseconds.

A future position is now calculable by setting a value  $t$  e.g. 1000 to predict the position which will be reached within the next second.

### 4.2 Simulation tool

For realizing and testing our algorithm we created a simulation tool called PlaSim. It consists of the following modules:

- The network module provides classes to emulate mobile devices and leaf servers.
- The scenario database stores simulation files recorded from ANSim (see below)
- The visualization module provides classes of the GUI (menu and graphical simulation output)
- The evaluation module contains all functions needed to analyze data recorded during PlaSim runs and a database where the predicted positions are stored.



**Figure 3: Snapshot of PlaSim during a simulation run. The shown environment is a model of our office area. Each number represents a simulated mobile.**

Figure 3 depicts a snapshot of PlaSim’s graphical user interface. PlaSim is implemented in Java so it could be easily integrated into PLASMA (see figure 4). It uses the PLASMA sighting proxy for managing the sighting information. For position prediction the server needs to know more than one sighting of every object. For this reason a new class that stores and manages multiple sightings for a single mobile was implemented and integrated into the sighting proxy. We used the Ad-Hoc network simulator ANSim [5] to generate the movement data needed for the simulation. In our set-up it uses the random waypoint model and we described our own office as the area of allowed positions (see figure 3). The movement data generated by ANSim was stored in the scenario database and used as input for PlaSim.

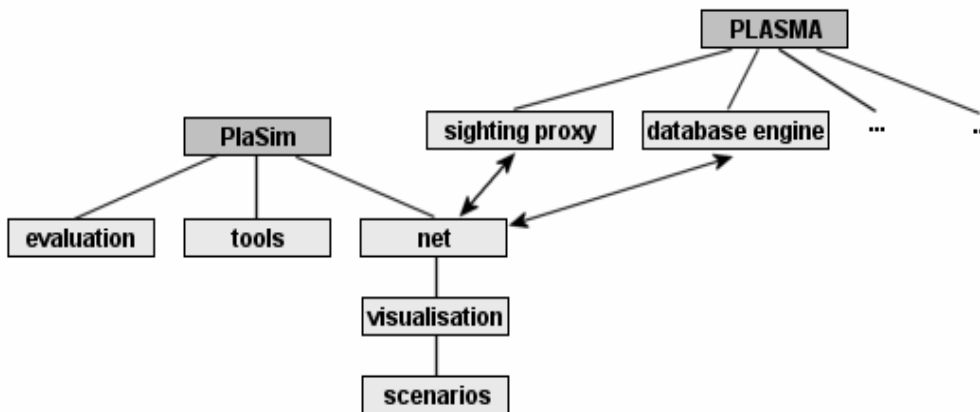


Figure 4: Integration of PlaSim and PLASMA.

The number of mobiles and leaf servers is fixed during a single simulation of the program, furthermore for the measurements there was just one leaf server for the office area. The here used leaf server is not a real platform server but provides basic functions, e.g. registering objects, receiving sighting information and the possibility to check if a mobile is still inside the aura of the server. The simulation runs in real time. This is more convenient due to the fact that real time stamps are needed for the prediction algorithm. Before starting the simulation PlaSim loads the position information from the scenario database to re-compute the movement of the objects during the simulation. The predicted positions are stored in the database of the evaluation module.

The evaluation module calculates the distance between all predicted points and real position of the mobile at this point in time. Then it determines the mean and standard deviation of all values of a simulation run. In addition, the percentage of values with a distance lower than 10 centimeters, between 10 and 20 centimeters and so on are computed. For less than 0.5 per cent of the recorded values per simulation the difference between the predicted position and the real position was larger than ten meters. These values were eliminated before the mean and the standard deviation were calculated.

## 5 Simulation

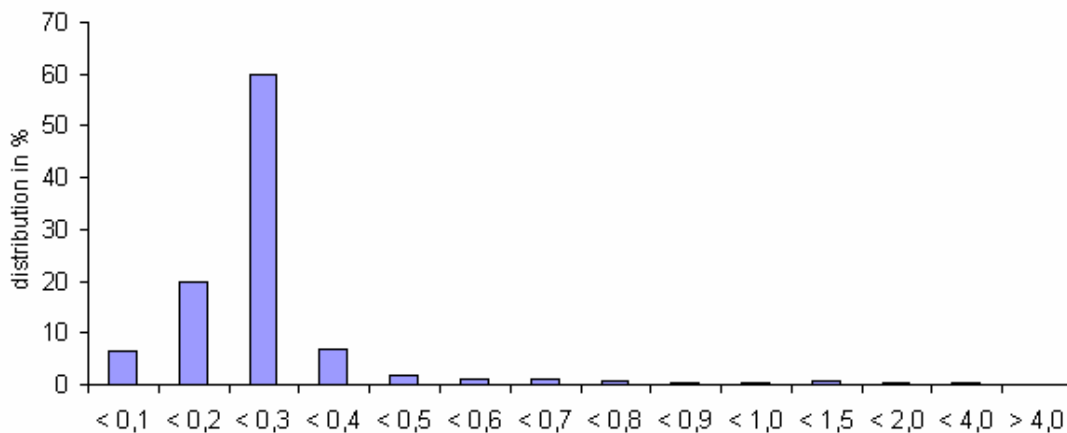
### 5.1 Simulation Set-up

At the start of the simulation a mobile first registers to the server where it is located. After this, every mobile starts to “walk” through the simulation area using the movement data stored earlier. During the movement the mobile sends every second its current position to the server. There the data are converted into geographical coordinates (with longitude, latitude and altitude). PlaSim uses the original geographical coordinates of our office to ensure realistic scenarios. Out of this position information the server can create PLASMA compliant sighting information. When receiving a sighting the server first forwards this to the sighting storage class and then calls its prediction method. In order to allow the evaluation of the

algorithm the predicted position is stored together with the real position in the evaluation database.

## 5.2 Measurement Results

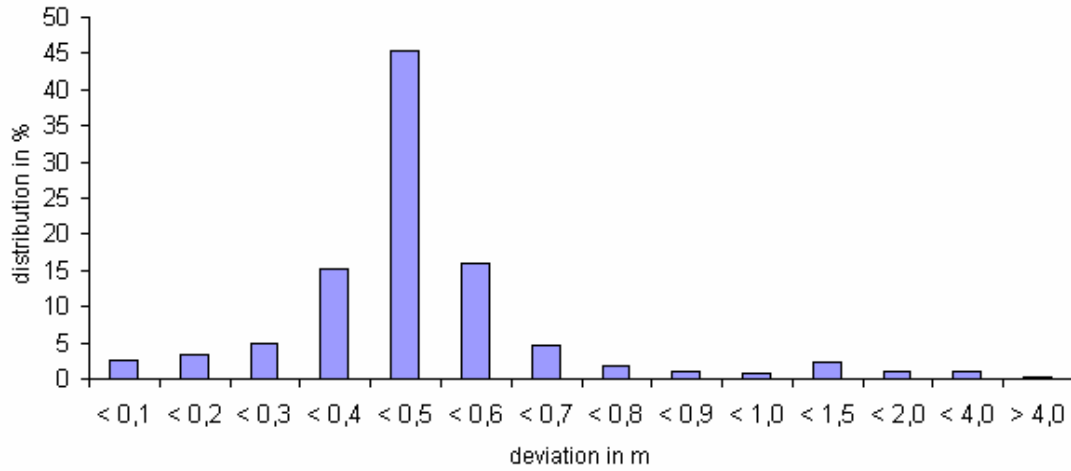
We simulated different prediction scenarios, i.e. the update rate of the position information was set to one and two seconds and the point in time for which a position was calculated was varied i.e. from one to three seconds. Every combination of update rate / prediction time was tested with seven different scenarios retrieved using ANSim as described above. The scenarios differ in the number of simulated mobile devices and of course in the movements of the mobiles. Every simulation resulted in several ten thousand predicted/real point value pairs, altogether about one million value pairs. After receiving the results of all scenarios of a update rate/prediction combination we determined the mean and the standard deviation for this combination.



**Figure 5: Evaluation results for a one second prediction in connection with a position update rate of one second.**

As expected the best prediction was made for a one-second prediction with a position update rate of one second. Those simulations reached a mean difference between real and predicted position of about 26 centimeters and a standard deviation of 20 centimeters. Figure 5 shows the average results of these simulations with the mentioned parameters. Note that more than 85 % of the values have a distance of less than 30 centimeters and only for 6 % the difference is larger than 50 centimeters. The results are worse for predictions e.g. with a time period of two or three seconds. For a prediction of two seconds the most value pairs, i.e. about 60 %, see figure 6, have distance of 30 to 50 centimeters.

The significant difference in the quality of the prediction in dependence of the prediction time led us to the assumption that a similar temporal distance between past and future may provide better results. The simulations with an update rate of two seconds and a two second prediction (see table 1) confirmed that assumption. Although the mean distance is much higher, still nearly 70 per cent of the values have a distance of less than 30 centimeters. For an update rate of one second and a prediction time of two seconds only ten per cent of the predictions have a



**Figure 6: Evaluation results for a two second prediction in connection with a position update rate of one second**

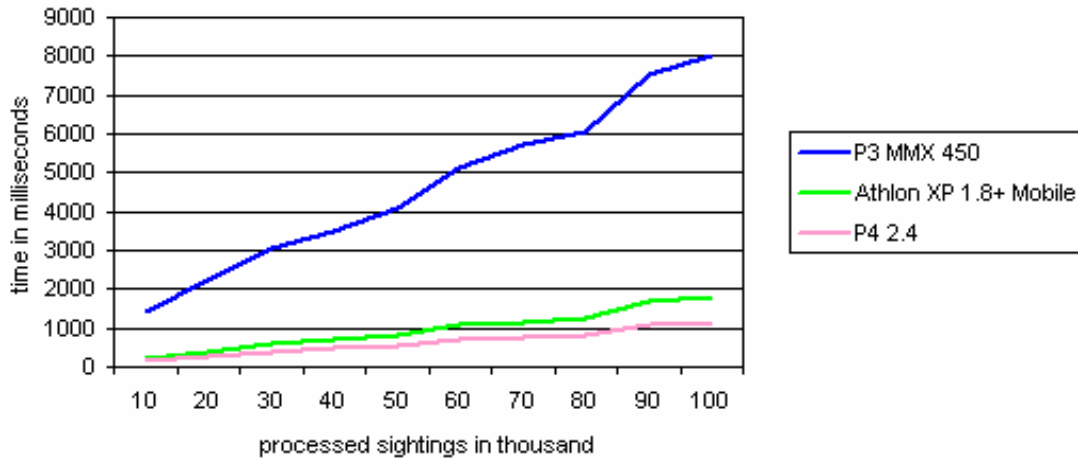
comparable accuracy. This behavior can be explained by the fact that with more frequent position updates more changes of the direction of the movements are detected. Thus, the prediction takes these slight changes into account which then leads to worse prediction results. In other words, a longer look into the past helps to determine the major direction more accurate. For long term predictions, e.g. three seconds, we recommend to use the current position as well as the position which was reached at least two seconds before the prediction was started.

	update rate / prediction time					
	1/1	1/2	1/3	2/1	2/2	2/3
mean in m:	0.26	0.50	0.69	0.29	0.43	0.50
values < 0.3 m in %:	86.11	10.60	4.94	75.77	69.33	37.37

**Table 1: Quality of the prediction for different parameters concerning update rate and prediction time.**

Our results show that the quality of the prediction depends highly on the freshness of the data used for the prediction i.e. the more current the position information is the better is the prediction. The second parameter that influences the prediction quality is the prediction time. The shorter it is the better are the results. This is due to the fact that in these cases the prediction system can adjust its prediction more frequently and with more accurate data. Since the overhead caused by the prediction is negligible we recommend to use the most frequent update rate of the position information and to do prediction only for the same time period. So a very good accuracy can be achieved.

In addition to the evaluation of the prediction quality we evaluated the performance of our approach. Our measurements showed that the delay introduced by the prediction is negligible.



**Figure 7: Performance comparison of prediction algorithm on different processors**

This is due to the relatively simple prediction mechanism. As figure 7 shows, the average time to process a single sighting including a prediction is far below a millisecond even on the slowest machine in the test (Pentium 3 with 450 MHz). Thus, the prediction does not cause any significant performance degradation, but may help to improve the performance of a leaf server in case of several handover operations per second.

## 6 Conclusion and outlook

We have introduced a simple position prediction mechanism that can be used without any kind of training phase and which is therefore well suited for hot spots with a lot of one time users. Our simulations have shown that our approach delivers very good results with respect to the accuracy of the prediction as well as with respect to the time needed per prediction. In order to initialize a handover operation it is sufficient to know the area in which the user will be within the next seconds. Thus, even with an position information update rate of three or more seconds our mechanism will still be applicable. In addition most applications, at least those we have in mind, do not require one hundred per cent accuracy.

In order to analyze the benefits of our approach in case of handovers in more detail we are going to extend our simulation environment with additional leaf servers and a load profile for the servers. In our next research steps we will investigate how our prediction mechanism can be used to improve the behavior of clients in case of disconnected operation. E.g. it will enable the platform to push additional data to the client, so it can provide at least some simple services, e.g. providing some information on a certain part of the city.

## Acknowledgements

This work was partially funded by the German government under grant 01AK044C.

## References

- [1] P. Langendoerfer, O. Maye, Z. Dyka, R. Sorge, R. Winkler, R. Kraemer: *Middleware for Location-based Services: Design and Implementation Issues*, in Q. Mahmoud (ed) *Middleware for Communication*, Wiley, to appear spring 2004
- [2] A. Patterson, R. R. Muntz, C. M. Pancake: *Challenges in Location-Aware Computing*, IEEE Pervasive Computing, Summer 2003.
- [3] F. Erbas, K. Kyamakya, J. Steuer, K. Jobmann: *On the user profiles and the prediction of user movements in wireless networks*, 0-7803-7589-0/02 IEEE, 2002
- [4] J. Biesterfeld, E. Ennigrou, K. Jobmann: *Neural Networks for Location Prediction in Mobile Networks*, Institut für Allgemeine Nachrichtentechnik, Universität Hannover, 1997.
- [5] H. Hellbrück, S. Fischer: *Towards Analysis and Simulation of Ad-Hoc Networks*, in Proc. of the Int. Conf. on Wireless Networks (ICWN'02), Las Vegas, USA, IEEE Computer Society Press, 2002.