PARAMETRIC DIMENSIONING AND ENHANCEMENTS OF A PATH PREDICTION ALGORITHM

Miltiadis Kyriakakos
Department of Informatics and Telecommunications,
University of Athens
Athens, Greece

Stathes Hadjiefthymiades
Department of Informatics and Telecommunications,
University of Athens
Athens, Greece

Lazaros Merakos
Department of Informatics and Telecommunications,
University of Athens
Athens, Greece

ABSTRACT

To meet the growing requirements for resource management schemes in the context of mobile and wireless computing, path prediction algorithms are gaining more attention and are being extensively examined. Path prediction allows the network to enhance the user quality of service. A parametric dimensioning of a path prediction algorithm based on learning automata is presented. The algorithmic parameter values are optimised for maximum prediction accuracy. Furthermore, two add-on mechanisms of the path prediction algorithm are introduced, and their performance is evaluated through simulations.

I. INTRODUCTION

The occurrence of handovers in cellular mobile networks is a very important issue, and constitutes the main research drive for the design of resource management mechanisms for such systems. Network and service engineers are mostly concerned with the elimination of the problems induced by the disruptions in physical connectivity & service provision. During the recent past, the pro-active method, involving movement (path) prediction, has been adopted for overcoming such problems in many diverse aspects of mobile computing. Pro-active management of network resources, driven by a properly structured movement estimator, enables the network designer to bypass (instead of correcting) the negative side effects of handovers. Network resources that could be managed through this scheme include (but are not limited to) bandwidth, MAC frames, files and packets. The network mechanisms, acting prior to the occurrence of the handover, may reserve bandwidth or MAC frames, relocate files or packets to the most likely to be visited neighboring cell. After the occurrence of the handover, the mobile terminal does not compete for such finite network resources (i.e., the bandwidth capacity in the new base station is limited; the same applies to the file storage area or the buffer capacity) but enjoys a pre-arranged configuration. Hence, due to the pro-active management of network resources, the nomadic user does not experience service discontinuity (increased drop probability) or performance degradation. Such a proactive resource management scheme is mostly meaningful in wireless-mobile networking infrastructures (and services) with relatively small cell residence times and multiple handover occurrences during typical user sessions. Wireless LANs are indicative examples of such infrastructures.

In this paper, we present a parametric dimensioning of a path prediction algorithm (PPA) initially discussed in [11] and [15]. The considered algorithm is based on a technique from Artificial Intelligence (AI), namely a Learning Automaton (LA) [12]. The LA, based on a properly structured knowledge base, assigns probability values to the neighbourhood of the currently visited cell. The highest probability denotes the most likely to be visited cell. In this paper, two add-on mechanisms are also introduced for enhancing the performance of the above scheme. The performance assessment of the above procedures has been based on a fully controllable mobility pattern generator platform called RMPG (Realistic Mobility Platform Generator) [16].

The rest of the paper is structured as follows. In Section 2, we discuss relevant prior work. In Section 3, we briefly discuss the details of a LA-based prediction scheme. In Section 4, a parametric dimensioning is presented. Two modifications of the prediction scheme are presented in Section 5. Finally, conclusions and results are included in Section 6.

II. PRIOR WORK

In this section, we briefly present other research efforts on the issue of predicting the movement of a nomadic user and pro-actively performing resource management.

In [6], the authors use the history of handovers to predict the next cell. The information of handovers is stored in a profile server, which translates historical data into mobility profiles. Such profiles represent the mobility pattern of each user. At each base station, and for every user, a next-cell prediction algorithm is executed. The algorithm informs the predicted neighbour cell and triggers advance resource reservation. When a user performs a handover from cell c′ to cell c, the user's profile is used to predict the next cell. Algorithm's decisions are based on previous history of user handovers from cell c′ to cell c″ through cell c. If such information is not available, the cell's profile, built on the cached information for the most recent handovers of all users, is used to predict the next cell for the specific user.

A probabilistic model of the history of handover behaviour is proposed in [7]. The model is based on the aggregate history of all handovers that occurred in a given cell. Two stages are foreseen, namely, the handoff estimation and the predictive-adaptive bandwidth reservation. In the first stage, that falls in the scope of this paper, each BS, involved in handovers, caches quadruplets in the form (Tevent, prev, next, Tsoj) for a roaming terminal. Such entries are called "hand-off event quadruplets". Tevent is the time when the terminal departed from the current cell, prev is the index of the previous visited cell, next is the index of the next cell, Tsoj is the cell sojourn (residence) time of the terminal. From the cached quadruplets, the BS builds a handoff estimation function (HOE), which describes the estimated distribution of the next cell and sojourn time of a mobile, depending on the cell the mobile came from.

In [8], the mobility tracking problem in a cellular network has been considered in the information theoretic framework. Comparison of user mobility models has been based upon the concept of entropy. A dictionary of user's path updates is built and maintained by the proposed scheme. Such dictionary supports an adaptive online algorithm that learns the profiles of subscribers. This technique is based on ideas and concepts coming from the area of lossless compression, and the Lempel-Ziv algorithm in particular. The algorithm is called "LeZi-update" and is exploited to reduce the location update related costs while its predictive power is used to reduce paging cost. In [9], the LeZi-Update scheme is applied in an intelligent home-environment in order to track down an inhabitant, both inside and within surroundings, so as to satisfy connectivity requirements.

The algorithm discussed in [10] is based on Mobile Motion Prediction (MMP) scheme for the prediction of the future location of a roaming user according to his movement history patterns. The scheme consists of Regularity-Pattern Detection (RPD) algorithms and Motion Prediction Algorithm (MPA). Regularity Detection is used to detect specific patterns of user movement from a properly structured database (IPB: Itinerary Pattern Base). Three classes of matching schemes are used for the detection of patterns namely the state matching, the velocity or time-matching and the frequency matching. The Prediction Algorithm (MPA) is invoked for combining regularity information with stochastic information (and constitutional constraints) and thus, reach a decision - prediction for the future location (or locations) of the terminal.

The work presented in [1] uses pattern matching techniques and Extended, Self Learning, Kalman filters to estimate the future location of mobile terminals and, thus, perform advance resource reservation and optimal route establishment in ATM based architectures. User Mobility Patterns (UMB) are stored in a database and fed to an approximate pattern matching algorithm to allow estimation (Global Prediction, GP) of a terminal's inter-cell movement direction (deterministic model). The Kalman estimator deals with the randomness in user movement by tracking intra-cell trajectory (stochastic model - Local Prediction, LP). The two models are combined together (Hierarchical Location Prediction) for the derivation of a semi-random movement trajectory. Simulation of the algorithm has shown that it accomplishes a high degree of prediction accuracy as soon as the Kalman filter becomes stable.

In [14], two mobility tracking algorithms, MT-1 and MT-2, are proposed based on RSSI measurements. The overall structure of the proposed MT-1 mobility estimator is consisting of a pre-filter, a modified Kalman filter, and an extended Kalman filter. The pre-filter consists of an averaging filter and a coarse position estimator. The pre-filter outputs a vector of position estimates that are used as the observation data for the modified Kalman filter. However, the accuracy of the mobility state estimates is largely dependent on the performance of the pre-filter. In order to avoid any inaccuracy and error in the pre-filter, a second (extended) Kalman filter is introduced to produce the mobility state estimates. The extended Kalman filter takes the averaged pilot signal strengths as observations and the estimated discrete command states from the modified Kalman filter to generate the estimates for the mobility states. If it is not possible to collect the observation data then the pre-filter cannot be used. In such a case MT-2 algorithm is used, where the raw observation data are passing through the extended Kalman filter.

In [13], a real-time mobility estimation scheme based on a first-order autoregressive model is presented, referred to as the AR-1 model. Based on that model an autoregressive position estimator is introduced. AR-1 position estimator consists of a mobility state estimator and a model parameter estimator. The estimation of mobility state is succeeded through the use of signal measurements typically available in wireless networks, i.e., received signal strength indicators (RSSI) or time of arrival (TOA) of a signal. Three independent signal measurements of either kind can be applied as observations to an extended Kalman filter in order to estimate the mobility state of a user.

Further work in the area has been reported in [3], [4] and [5]. In [3] and [4], direction information and departure history are used to predict the future cell/location of a mobile unit. In the absence of history, a supporting mechanism, termed Correlation Criterion, is invoked by monitoring the movement patterns of the other users. In this way, a departure history is obtained for prediction purposes. In [5], the mobile unit executes a flexible prediction algorithm. The mobile keeps historic information of its trajectory and compares it with existing mobility patterns maintained in a database. For the matching process, Genetic Algorithms are used.

III. PATH PREDICTION ALGORITHM

Artificial intelligence has been used in [11] in order to predict the next cell for a terminal. Specifically, a learning automaton has been used. It is based on a state transition matrix, which comprises the one-step state transition probabilities and follows a Linear Reward-Penalty (L_{R-P}) scheme. If the automaton's decision is correct, a positive feedback is received from the environment and the probability of the respective state transition is increased ("rewarded"). The rest of the probabilities are evenly reduced, "penalized" in order to keep the probability sum equal to one. If the response is wrong the state transition is "penalized" and the rest of the transitions are "rewarded" accordingly. This behaviour is shown in equation (1). The path prediction algorithm is executed at the home registry of the terminal. There is an itinerary database for each user with entries holding spatial and temporal information. When prediction is requested a set of entries are examined and the one with the highest probability is chosen. If that response is correct or not then the procedures mentioned above are invoked. Should no relevant entries be found in the database, new entries are introduced and a random decision is taken.

$$\begin{aligned}&\text{Transition } (i \rightarrow j) \text{ received positive feedback}: \begin{cases} P_{ij} = P_{ij} + w(1 - P_{ij}) \\ P_{ik} = P_{ik} \cdot (1 - w), k \neq j \end{cases} \\ &\text{Transition } (i \rightarrow j) \text{ received negative feedback}: \begin{cases} P_{ij} = P_{ij} - w'(1 - P_{ij}) \\ P_{ik} = P_{ik} \cdot (1 + w'), k \neq j \end{cases}\end{aligned} \quad (1)$$

In (1), w denotes the rewarding step value while w' denotes the penalizing step value. Those two values may be equal (a successful decision is equally important than an unsuccessful decision), or different (e.g., success is more important). In the simulation, w and w' assumed different, but constant, values. Generally, small step values for the reward/penalty procedures facilitate the automaton convergence to the correct actions [2], [12].

A variant of the above scheme proposed in [15] and operates as follows. Two learning automata, the Global (GA) and the Per-User (PUA), operate simultaneously on top of two itinerary databases for some specific users. Both GA and PUA are based on the reward/penalize procedure of (1). A scheme that combines the outcomes of both automata (PUA and GA) is referred to as Weighted Automaton (WA). When a prediction is requested, for a specific user, both the PUA and GA are consulted. The WA combines the per-cell probabilities (for all future cells) returned by the two automata using predefined weights, as defined in (2).

$$\begin{aligned}&P_{WA}(pc, cc, fc, ts, uid) = \beta \cdot P_{GA}(pc, cc, fc, ts) + \\ &(1 - \beta) \cdot P_{PUA}(pc, cc, fc, ts, uid), \quad \beta = 1/3\end{aligned} \quad (2)$$

In (2), P_{WA}(*pc*, *cc*, *fc*, *ts*, *uid*) and P_{PUA}(*pc*, *cc*, *fc*, *ts*, *uid*) denote the transition probability from (previous cell *pc*, current cell *cc*) to future cell *fc* for a given user *uid* and time-slot *ts*, for the WA and the PUA respectively. P_{GA}(*pc*, *cc*, *fc*, *ts*) denotes the transition probability from (previous cell *pc*, current cell *cc*) to future cell *fc* for a given time-slot *ts*. The reward/penalty procedure is always governed by (1) and applied independently to the PUA and the GA.

IV. PPA – PARAMETRIC DIMENTIONING

In this section, we examine the various parameters that influence the effectiveness and performance of the path prediction algorithm presented in the previous section. Subsequently, we defined the parameters that play decisive role for the performance of the GA, PUA and WA. Finally, after extensive simulations and trials of various values in the critical parameters we mention our results. From equations (1) and (2) we lead to the conclusion that the parameters that need to be evaluated are w' and β.

A. Weight w′

In (1), the dependence between the reward and the penalty steps is given by equation (3):

$$w = w' \cdot n, \ n = 1,2,...,5 \quad (3)$$

In (3), n is the indicator for the possible transactions form one CID to its neighboring cells apart from the one selected as a prediction output from the path prediction algorithm. The value of w, in the more general case (n = 5), is equal to 0.1 ([15]) while in the case where a one-step transition probability P_{ik} has a value smaller than 0.02 then it is not taken into consideration in the automaton's process of positive reward of P_{ij}. The logic of equation (3) shows that the proportion of w and w' is maintained while the possibility of a one-step transition probability to recover is always possible. Learning Automaton algorithms [17] prevent the reduction of a probability's value under a threshold value.

B. Weight β

β is the mixture factor of Global and Per User Automata in (2). We set the term (1-β) as B. B will be used for the dimensioning of weight β in the experiments that will follow.

C. Values of w′ and B

Simulations have been performed for a large number of (w', B) pairs. Several modifications took place in the platform to facilitate the simultaneous study of the joint behaviour of w' and B.

Initially, the one-step transition probabilities are set to 1/6. We keep w' at values not greater than 0.1, to avoid exceptionally abrupt fluctuations of the probabilities, which may adversely affect the convergence of the automata. Therefore, w' is examined in the interval (0, 0.1] as follows: in interval (0, 0.0025] with step value 0.00025, in the interval (0.0025, 0.01] with a step value of 0.0005 and in the interval (0.01, 0.1] with step value 0.0025. Overall, 61 values have been studied.

Respectively, for B we have examined the interval (0, 1) with step 0.1 (the values 0 and 1 correspond to the exclusive use of PUA and GA, respectively [10]). The values 0.33 and 0.66, which correspond to β=2/3 and β=1/3, respectively, have also been considered in the tests. We should note that the value B=0.66 has been selected in the initial presentation of WA [15].

Finally, after having performed several simulations of the path prediction algorithm, the values of the examined parameters have been narrowed for w' in the interval (0, 0.1] with step 0.0025, while for B in the values: 0.2, 0.3, 0.5, 0.75 and 0.6667. All the possible combinations of the aforementioned sets of values have been tested to provide the best-fit values of the examined parameters.

1) The parameter w′

In Figure 1, we present the highest values of prediction accuracy that GA achieves for all (w', B) pair combinations. Only combinations that gave prediction success higher than 62% (42 combinations) are presented. These combinations have values of w' in the interval [0.035, 0.04], while an important number of such combinations is also observed for values of w' higher than 0.07. A large number of 'good' values in the first interval is as expected. For example, for w'=0.035 we obtain w=0.17 (n = 5). Consequently, a successful prediction could more than double the corresponding probability, while the remaining five probabilities are decreased with a small step. In this particular case, one of the six state transition probabilities is strengthened after a a sufficient number of automaton calls (automaton convergence) in the particular cell, and simultaneously the possibility of recovering for the rest probabilities is maintained. For small values of w' we get a quite smooth behaviour of the automaton without intense oscillations of the state transition probabilities values. In the case where w'=0.7 (w=3.5), the automaton will converge after two similar successes. Consequently, the results of the automaton are very unstable (rapid changes) for large values of w'. We consider that the value w'=0.02, selected initially in [15], provides very satisfactory results.

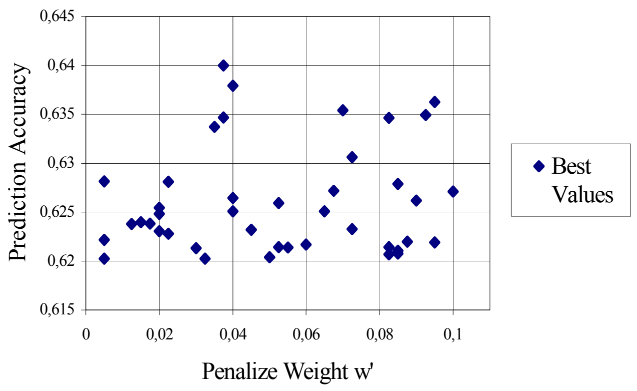

Figure 1: Prediction Accuracy for all (w', B)

2) The parameter B

In Figure 2 the distribution of combinations with percentage success higher than 63% (52 combinations) for values of B is presented. The higher values as well as the higher concentration of values are observed for the value 0.667 (B=2/3, β=1/3). Hence, the value β=1/3, selected initially in [15], provides quite satisfactory results.

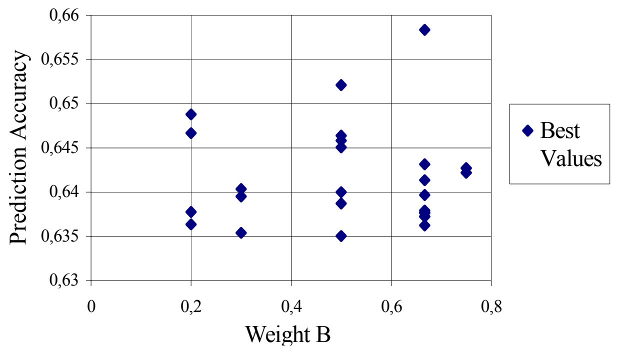

Figure 2: Percentage of Successful Predictions for B

V. PPA – ADD ON MECHANISMS

In this section, two add-on mechanisms for improving WA's behaviour are proposed.

A. Modified Mechanism(MM)

In the initial scheme, apart from the first best first prediction of WA, a record and measurement of the best second and third predictions (more likely next place) are considered.

More specifically, the second and third predictions are exploited in case the first prediction is not correct. Thus, if the first choice is erroneous, PPA is not being applied but the second best prediction is examined. The mechanism of reward/penalty is activated in the same way as for the first best prediction. In case the second best prediction is also not successful, PPA is not applied and the third best prediction is examined. By performing the previous controls, the automaton examines 50% or higher of the possible transactions from a cell.

B. Self-adjusting Mechanism(SAM)

WA has been adopted for smoothing out the effect of stochastic users. Depending on the behaviour of each user, WA performs better or worse than GA. Observing the results from [15] lead us to the following conclusions:

a) For users with high prediction accuracy the contribution of GA (small β) should be smaller than the contribution of PUA (large β).

b) For users with low prediction accuracy the contribution of GA (large β) should be bigger than the contribution of PUA (small β).

Based on the previous conclusions and equation (2) a less stochastic user should help a quite stochastic user. A dynamic behaviour of the WA is adopted, referred to as self-adjusting mechanism, where for each user a different weight β is adopted. Although, it is not possible to know beforehand the stochastic character of a user, the self-adjusting mechanism is adopted dynamically during the simulation of the users. In practice, the automata results are kept in a database. Specifically, after the 5th week (e.g., after the 5th week the automata reach 80% of their final prediction output) and at the end of each subsequent week the percentages of prediction accuracy achieved for a particular user (PUA) are read and compared with the percentages of GA's prediction accuracy (Equation (4):

$$D = \left| AP_{GA} - AP_{PUA} \right| \qquad (4)$$

In (4), AP_{GA} and AP_{PUA} denote the percentage of prediction accuracy of GA and PUA, respectively. When the difference, D, is greater than a specific value (e.g., greater than 0.1) then the value of β is increased. The mechanism of changing β follows:

$$\begin{aligned} & if\,D > 0.1 \Rightarrow \beta = \beta + \alpha, \;\; \beta_{\max} = 2/3, \quad a = \frac{\frac{2}{3} - \beta}{2} \\ & if\,D < 0.1 \Rightarrow \beta = \text{constant} \end{aligned} \qquad (5)$$

In (5), α is a constant added to β for D >0.1 and $\beta_{\max}$ is the maximum value β could achieve. β is initially set at the value of 1/3. Clearly, equation (5) exhibits an asymptotic behaviour for β, as the target value is never reached. In practice, after the fifth subsequent increase of β's value (β~0.6667) we set β equal to 0.6667. When the value of β becomes equal to 0.6667, it is kept at that level till the end of simulation.

VI. CONCLUSIONS

Figures 3, 4 and 5 present the WA results of the initial algorithm, the modified algorithm and the self-adjusting initial algorithm for the same three user profiles examined in [15], A, B and C, respectively. We observe that the modified and the self-adjusting mechanisms give an improvement of about 10% compared with the initial results of WA. We should note that although complexity has been increased compared with the initial algorithm, the simulation has not shown significant delay increase.

Another important general observation is that the two add-on mechanisms of the algorithm not only provide better output than the initial algorithm but they do so consistently. More specifically, even though the two proposed versions have been tested for fewer combinations of (*w′*, b) than the initial algorithm (420 tests for the initial, 200 for the modified and 41 for the self-adjusting algorithm), they have given more combinations with high percentage of prediction accuracy (e.g., for success percentage equal to 64%, the initial algorithm has produced 9 combinations (2%), the modified 163 combinations (81.5%), and the self-adjusting 34 combinations (83%)).

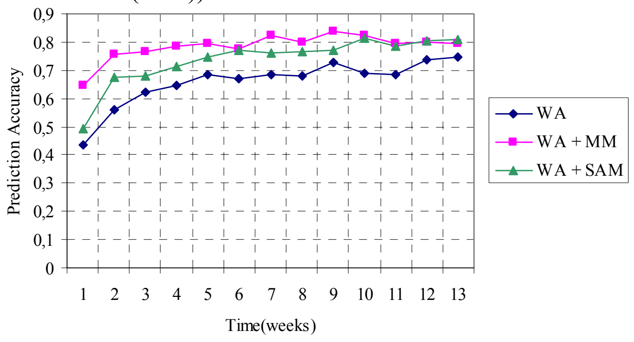

Figure 3: Profile A – Comparison of WA – Variations

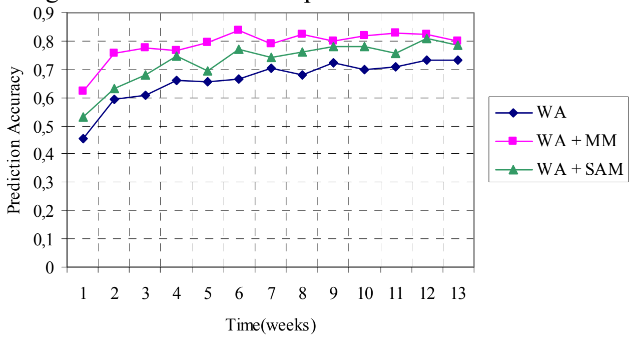

Figure 4: Profile B – Comparison of WA – Variations

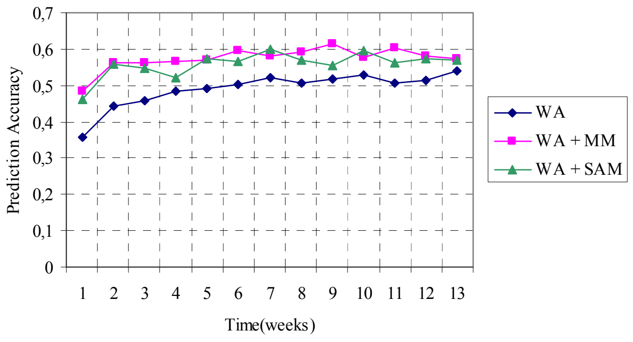

Figure 5: Profile C – Comparison of WA – Variations

REFERENCES

[1] T.Liu, P.Bahl, and I.Chlamtac, “Mobility Modeling, Location Tracking, and Trajectory Prediction in Wireless ATM Networks”, IEEE JSAC, Vol.16, No 6, August 1998.

[2] M.A.L.Thathachar, “Learning automata Processing Ergodicity of the Mean: The Two-Action Case”, IEEE Transactions on Systems, Man, and Cybernetics, Nov./Dec. 1983.

[3] J.Chan, S.Zhou and A.Seneviratne, “A QoS Adaptive Mobility Prediction Scheme for Wireless Networks”, Proceedings of IEEE GLOBECOM'98, Nov. 1998.

[4] J.Chan and A.Seneviratne “A Practical User Mobility Prediction Algorithm for Supporting Adaptive QoS in Wireless Networks,” Proceedings of ICON’99, Brisbane, 1999.

[5] L.Q.Liu, A.T.Munro, and M.H.Barton, “Efficient Mobility Management: A New Flexible Design Algorithm”, proceedings of ICUPC ’96, Cambridge, MA, USA, September 1996.

[6] V.Bharghavan, and M.Jayanth, “Profile-based Next-cell Prediction in Indoor Wireless LAN,” Proceedings of IEEE SICON’97, Apr. 1997.

[7] S.Choi, and K.G.Shin, “Predictive and Adaptive Bandwidth Reservation for Hand-offs in QoS-Sensitive Cellular Networks,” Proceedings of ACM SIGCOMM ‘98, Vancouver, British Columbia, September 1998.

[8] A.Bhattacharya, and S.K.Das, “LeZi Update: An Information Theoretic Approach to Track Mobile Users In PCS Networks,” Proceedings of ACM/IEEE Mobicom ’99, Seattle, USA, August 1999.

[9] S.K.Das, D.J.Cook, A.Bhattacharya, E.O.Helerman and T-Y.Lin, “The Role of Prediction Algorithms in the MavHome Smart Home Architecture”, IEEE Wireless Communications, vol. 9, no. 6, pp. 77-84, December 2002.

[10] G.Y.Liu, G.Q.Maguire Jr. “A Class of Mobile Motion Prediction Algorithms for Wireless Mobile Computing and Communications”, MONET, Vol.1, pp.113-121, 1996.

[11] S.Hadjiefthymiades, and L.Merakos, “Proxies + Path Prediction: Improving Web Service Provision in Wireless Mobile Communication”, ACM/Kluwer MONET, Vol.8, Issue 4, August 2003.

[12] K.Narendra, and M.A.L.Thathachar, “Learning Automata: An Introduction”, Prentice-Hall, 1989.

[13] Zaidi, Z. R. and B. L. Mark, “Real-time Mobility Tracking Algorithms for Cellular Networks Based on Kalman Filtering,” IEEE Transactions on Mobile Computing, 2005, Vol. 4 , no. 2, pp. 195 – 208, March/April 2005.

[14] Zaidi, Z. R. and B. L. Mark, “Mobility Estimation for Wireless Networks Based on an Autoregressive Model,” Proceedings of IEEE Globecom ‘04, Dallas, Texas, December 2004.

[15] Kyriakakos, M., S. Hadjiefthymiades, N. Frangiadakis and L. Merakos, “Enhanced Path Prediction for Network Resource Management in Wireless LANs,” IEEE Wireless Communications Magazine, Special issue on The Evolution of Wireless LANs and PANs, vol. 10, no. 6, pp. 62–69, December, 2003.

[16] Kyriakakos, M., N. Frangiadakis, S. Hadjiefthymiades and L. Merakos, “RMPG: A Realistic Mobility Pattern Generator for the Performance Assessment of Mobility Functions”, Journal of Simulation Modeling Practice and Theory (SIMPAT), Elsevier Science, vol. 21, issue 1, pp. 1–13, 2004.

[17] Friedman, E. and S. Shenker, Synchronous and asynchronous learning by responsive learning automata. MIMEO, 1996.