Using Proxy Cache Relocation to Accelerate Web Browsing in Wireless/Mobile Communications

Stathes Hadjiefthymiades
University of Athens, Dept. of Informatics and Telecommunications
Panepistimioupolis, Ilisia, Athens, 15784, Greece
Tel.: +301 7275334
shadj@di.uoa.gr

Lazaros Merakos
University of Athens, Dept. of Informatics and Telecommunications
Panepistimioupolis, Ilisia, Athens, 15784, Greece
Tel.: +301 7275323
merakos@di.uoa.gr

ABSTRACT
Mobile computing is considered of major importance to the computing industry for the forthcoming years due to the progress in the wireless communications area. A proxy-based architecture for accelerating Web browsing in cellular customer premises networks (CPN) is presented. Proxy caches, maintained in base stations, are constantly relocated to accompany the roaming user. A cache management scheme is proposed, which involves the relocation of full caches to the most probable cells but also percentages of the caches to less likely neighbors. Relocation is performed according to a movement prediction algorithm based on a learning automaton. The simulation of the scheme demonstrates substantial benefits for the end user.

Keywords
Mobile Computing, W4, Proxy Cache, Cache Relocation, Path Prediction, Learning Automaton.

1. INTRODUCTION
Currently, the WWW [10] is viewed as a very promising technology and used vastly for the deployment of applications in the Internet and corporate intranets. This client/server system, conceived in the early 90's, owes its great success in the standardization of the communication between clients (browsers) and information servers. The three open standards that are primarily involved in such communication are the HyperText Transfer Protocol, the HyperText Markup Language, and the Universal Resource Identifiers addressing scheme.

Apart from the developments in the software area, during the 90's, we have also witnessed great advances in the area of wireless personal communications. The European cellular system GSM (Global System for Mobile Communications) [35] has received an unprecedented acceptance and spread rapidly over the globe. Office environments and small industrial installations have also benefited from the introduction of the DECT standard [18]. More sophisticated applications, enriched with multimedia capabilities (e.g., voice, VoD), have been made feasible with HIPERLAN (High Performance Radio Local Area Network) [23]. A number of ATM based wireless LAN prototypes have been recently developed and discussed in the wireless networking literature [34], [17]. The near future is also very promising: the introduction of the fully-fledged Universal Mobile Telecommunications System (UMTS) is planned for the period 2002 - 2005 [38]. This emerging standard will provide users with data rates up to 2 Mbps, circuit and packet switched service, while world wide coverage (through satellite, macro-, micro-, or pico-cell environments) is among the objectives of constantly progressing research in the area. UMTS, combined with technologies like WAP (Wireless Application Protocol) and VHE (Virtual Home Environment) [19], seems an ideal platform for mobile multimedia services. The growth of wireless telecommunications has stimulated the interest for the so-called anywhere - anytime computing. This type of computing, also known as "nomadic computing" [27], aims to provide users with access to popular desktop applications, applications specially suited for mobile users, and basic communication services. The emergence of nomadic computing has also been facilitated by the rapid proliferation of portable computing equipment (portable PCs, portable digital assistants).

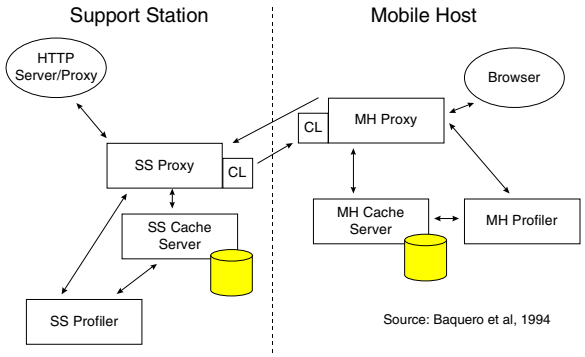

Figure 1. MobiScape architecture

During the past years, a number of efforts have been made to consolidate the WWW with wireless networks (such integration is also referred to as W4 - World Wide Web for Wireless). Notable examples of such efforts are the MobiScape [8] and the IBM Web Express platforms [25], [20], which are used as a basis for our work. Both systems capitalize on the proxy interface that most modern navigation tools support. The architecture developed in MobiScape is graphically shown in Figure 1.

WWW10, May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

The Mobile Host (MH) uses the Support Station (SS) as a gateway to the WWW. A caching mechanism is provided in both the MH and the SS in order to minimize the periods of connection between the two machines. The cache in the MH facilitates disconnected operation. The cache in the SS reduces the wait periods experienced when fetching remote documents. Thus, the HTTP data flow between the browser (MH) and the HTTP server is intercepted twice (in the MH and the SS). Both the MH and SS cache servers operate as proxies [33]. The data flow between the SS proxy and the MH proxy is compressed (CL: Compression Layer). The profilers, which support the operation of both proxies, consult user-defined scripts and trigger the pre-fetching of a series of documents that are very likely to be requested during the user's session. Novel ideas for a similar architecture (also termed "intercepting technology") are also proposed in the Web Express work. In addition to the above, the Web Express platform addresses the requirements of transaction processing in Web applications through the adoption of "differencing" techniques.

The MobiScape/Web Express work, which is well harmonized with existing Web software and does not require changes in browsers and servers, makes no provisions for the roaming of the mobile user. If the wireless infrastructure provides for handovers between base stations, the SS cache has to be reconstructed each time the mobile terminal crosses the boundaries of a cell and establishes connection with a new base station. In [22], we suggested an extension of the MobiScape/Web Express work. We proposed the constant relocation of the SS cache so that it follows the movement of the mobile station. Cache relocation is performed prior to the realization of handovers according to movement prediction algorithms. Such algorithms provide the means for pro-active resource management. The algorithm used takes into account the randomness in the movement of the user as well as the already identified movement patterns to reach combined decisions.

In this paper, we study a quite different cache management scheme: different portions of the original cache are relocated to all the cells being adjacent to the one currently used. A new algorithm for the movement prediction of the nomadic user is proposed and its performance is evaluated through simulations. Lastly, based on the simulation results of the path prediction algorithm, we evaluate the performance of the new cache management scheme. We show differences, in resource retrieval delays, from conventional architectures in the order of 22.2%.

The paper is structured as follows. Section 2 presents the considered architecture. Additionally, in Section 2 we discuss the "moving" cache approach. Such approach is based on a cache relocation scheme and a path prediction algorithm. The cache relocation scheme is presented in Section 3, where we also discuss performance issues related to WWW proxy caches. Our new path prediction algorithm is presented in Section 4. In Section 5, we discuss additional details of the simulation model. Specifically, we consider the modeling of WWW traffic and cell residence times. In Section 6, we provide the results of the simulation of the path prediction algorithm and the overall cache relocation scheme. Section 7 contains our conclusions.

2. SYSTEM ARCHITECTURE – THE "MOVING" CACHE APPROACH

Figure 2 illustrates the wireless Customer Premises Network (CPN) environment assumed in this study. The environment consists of a number of base stations (BS) interconnected by means of fixed LAN infrastructure, mobile and fixed terminals. Individual LANs are interconnected by means of routers; one of them is acting as an Internet gateway. Each BS comprises a radio transceiver and a support workstation. The latter deals with all the signaling needed for the roaming of the mobile terminal. In addition, the BS assumes the role of the Support Station encountered in MobiScape.

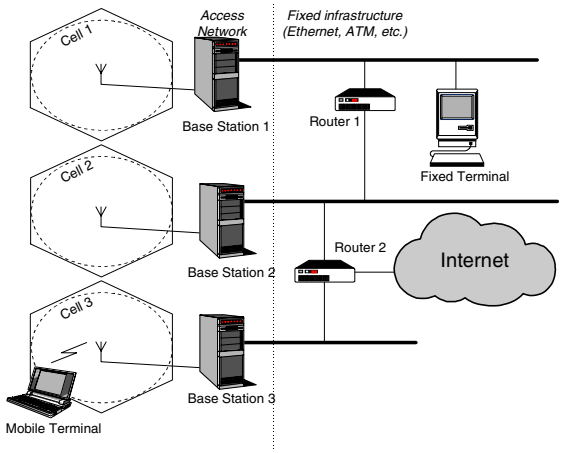

Figure 2. Network architecture

Wireless cells are assumed hexagonal and cover the entire surface of the installation (e.g., a university campus, a factory). BSs maintain information regarding the mobile terminals that are currently under their control. In the event of a handover, the involved BSs (current, target cell) need to consult their information base, and collectively undertake specific actions, such as reservation of bandwidth in the new path, release of resources in the old path, diversion of connections and addressing updates. In the majority of wireless architectures, user profiles are stored in specialized nodes within the user's home sub-network (the part of the network the user administratively belongs to). When the mobile terminal migrates to a sub-network different from its home, the user profile database (home registry) is queried and relevant information is forwarded to the visited network by means of specialized inter-network signaling [4].

The home registry of a mobile terminal may also incorporate a path prediction algorithm (Figure 3), similar to the one presented in the next paragraphs. Such an algorithm predicts, with adequate precision, to which cells the terminal is more likely to be handed-over if it keeps on roaming. The invocation of the path prediction algorithm can be performed at some time after the entrance of the mobile terminal in the current cell. From that point, the current BS relocates its cache (or parts of it) to the BSs indicated by the path prediction algorithm. The mobile terminal is likely to attach to those BSs (also referred to as Target BSs) in the near future. It is not wise to invoke the path prediction algorithm (and, thus, relocate the accumulated cache) before the above mentioned time period elapses, as the cache can still be augmented by user requests and, thus, better hit rates can be achieved. The sequence of actions undertaken by network entities as well as the inter-network signaling required for their completion are depicted in the Message Sequence Chart (MSC [26]) of Figure 4.

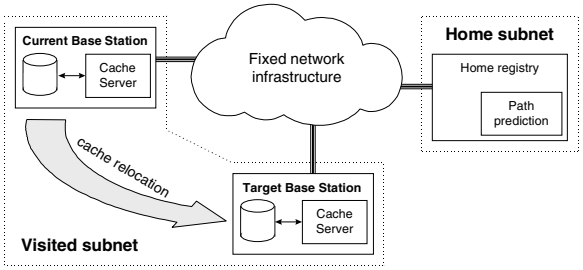

Figure 3. The "moving" cache technique

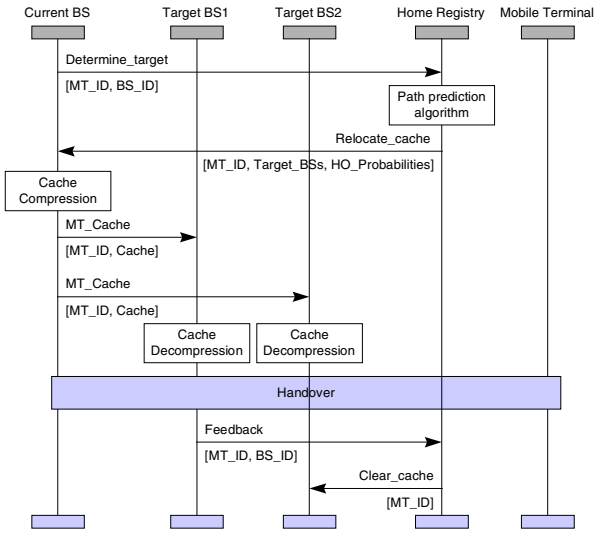

Figure 4. MSC for cache relocation and handover

In Figure 4, the **Determine_Target[MT_ID, BS_ID]** signal is used for triggering the path prediction algorithm in the home registry. Its parameters denote the identification of the MT for which the algorithm should be executed, as well as the identification of the current BS. The Home Registry invokes the path prediction algorithm and notifies, through the **Relocate_Cache[MT_ID, Target_BSs, HO_Probabilities]** signal, the current BS. The Target_BSs parameter is a list that contains the identifiers of all the neighboring cells (to the one currently used). The HO_Probabilities parameter is also a list that assigns probability values to the items of the Target_BSs list. Practically, such probabilities denote the likelihood of the MT's entrance in the respective cells. The current BS relocates its cache, or parts of it, to a subset of the neighboring BSs according to the probabilities found in the HO_Probabilities list. This scheme will be discussed in more detail in subsequent paragraphs.

The relocation of the cache is realized by means of the MT_Cache signals shown in Figure 4. The cache segments contained in those signals have been previously packed and compressed by the current BS. The information is de-compressed by the target BSs and fed into their local cache. It should be stressed that the whole procedure is executed in parallel to MT's roaming, without obstructing its communication through the current BS. At some time after the relocation of its cache, the MT executes a handoff. The BS to which the MT has actually been handed over notifies the Home Registry through the **Feedback[MT_ID, BS_ID]**. As discussed below, such feedback is essential for the operation of the path prediction algorithm. The Home Registry that has complete knowledge of which BSs have received MT_Cache signals, notifies them, through the **Clear_Cache[MT_ID]** message. Clear_Cache triggers the removal from BS caches of those items pertaining to the designated MT.

3. CACHE RELOCATION SCHEME. PERFORMANCE ISSUES

In this section, we evaluate the performance of a cache relocation scheme which involves transmitting the entire cache from the current BS to the most likely Target BS indicated by the path prediction algorithm (i.e., the BS that corresponds to the highest probability value in the HO_Probabilities list). Furthermore, to deal with the cases of prediction misses, the scheme involves the transmission of the most popular segment (a 70%) of the accumulated cache to two of the other neighboring cells (the second best predictions of the algorithm). Even lower percentages (i.e., 30%) are relocated to the remaining neighboring cells. Such segments of the original cache are referred to as partial caches. This approach is quite similar to the Shadow Cluster [28] technique proposed for bandwidth reservation in wireless ATM. In the Shadow Cluster scheme, bandwidth is provisionally reserved in the cells adjacent to the one currently used by the mobile terminal.

Relocating the entire cache to all the adjacent cells is not a sound strategy, similarly to the bandwidth problem studied in [28], as it may lead to non-optimum use of limited resources in the involved BSs (i.e., the disk or bandwidth capacity of the candidate BS may be exceeded and thus, refuse the proxy service to other roaming users or force the release of their connections, respectively).

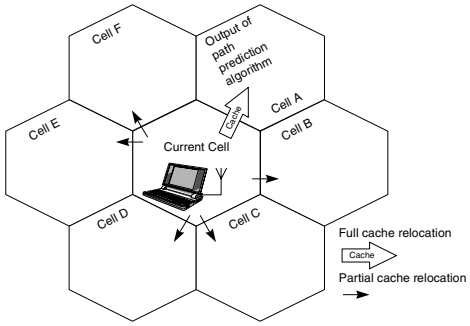

Figure 5. Full and partial cache relocation strategies

The use of a proxy cache within the access network is aligned with the general architecture suggested by the WAP Forum. WAP uses a proxy/gateway within the access network to convert formats and adapt exchanged traffic to something interpretable by the mobile terminal. A similar task could be assigned to our proxy server (functionality extension). We should also note the possibility of retrieving the requested resources, after the occurrence of a handover, from the previous BS. If the two BSs (previous and current) are attached to the same LAN, such an alternative could be worth adopting. The proxy within the current BS may relay incoming requests to the previous BS (much alike the path extension technique employed in Wireless ATM [2]). If, however, the old and the current BSs are not interconnected through the same LAN (see

Figure 2), then the suggested relocation scheme seems more suitable for the rapid dispatch of WWW requests.

One of the issues under study in this work is the performance achieved by the caching system of the BS proxies in the event of prediction misses. In the following paragraphs, we discuss how we have modeled the behavior of caches and partial caches. We have adopted a cache size of 2 MB per roaming user, which seems a reasonable volume of relocatable data. Based on measurements taken in our laboratory, we assume that a cache size of 2 MB can accomplish a hit rate of 17-20%. Such a hit rate was reported by the WinProxy software [39], configured to keep the local cache size under the 2 MB limit and accessed by a single HTTP client. When the WinProxy cache reaches its maximum configured size, a garbage collection mechanism is activated and the oldest cache objects are deleted first (cache replacement policy). The garbage collection mechanism brings the size of the cache down to the 85% of its maximum configured size. We have monitored the performance of the proxy system after a time period of 3 weeks (from system initialization) with regular workload (thus, we allowed the cache system to reach an equilibrium state and refine/filter-out the occasionally referenced resources from the really important ones). It should be stressed that the adopted cache hit rate accounts for a worst-case scenario (scenario where the cache accumulated for one user encompasses completely different files from that accumulated for another user - their intersection is the null set). Under regular system operation, the caches from two or more users may be mixed and, thus, accomplish hit rates higher than the one reported above. In any case though, the cache-hit rate cannot exceed the 50% ceiling reported in the WWW proxy literature [1].

The accumulated cache incorporates a set of WWW resources (HTML, images, etc.). The size of this set can be roughly estimated by considering the reported distribution for WWW resource sizes and the adopted total cache size of 2 MB. According to [9], the lognormal distribution provides the best fit for Web file sizes less than 133 KB (this family of files accounts for the 93% of the total set of files available in the WWW). The parameters of the lognormal distribution have been estimated at the following values: μ=9.357, σ=1.318. The ratio of the configured maximum size of the proxy cache (i.e., 2 MB) by the mean resource size (i.e., 9.357 KB) yields an approximate mean cache population of 220 items. The probability of accessing one of the resources stored in a proxy cache can be calculated through the Zipf distribution [21], [16], [9], [3], [5]. Specifically, the number of references to cached item i, NoR_i, satisfies (1):

$$NoR_i = \frac{a}{rank(i)^Z} \qquad (1)$$

In (1), *rank(i)* denotes the position of cached item i after the sorting of the cache population on the basis of references (i.e., the most popular item is ranked first, the second most popular is ranked second and so forth), a is a constant value while Z is the Zipf parameter. The Zipf parameter assumes a value close to 1; in [5], Z was estimated at 0.85, while in [16] a value of 0.986 was proposed for the parameter. In our simulation, we have adopted $Z = 0.85$.

Based on (1), we may induce the conditional probability, P_{hit}, of a cache hit in a partial cache containing the most popular part of the original cache, given a cache hit in the original cache, as follows:

$$P_{hit} = \frac{\sum_{i=1}^{\lfloor cache\_size \cdot C \rfloor} NoR_i}{\sum_{i=1}^{cache\_size} NoR_i} = \frac{\sum_{i=1}^{\lfloor cache\_size \cdot C \rfloor} \frac{a}{rank(i)^z}}{\sum_{i=1}^{cache\_size} \frac{a}{rank(i)^z}} = \frac{\sum_{i=1}^{\lfloor cache\_size \cdot C \rfloor} rank(i)^{-z}}{\sum_{i=1}^{cache\_size} rank(i)^{-z}} \qquad (2)$$

In (2), cache_size denotes the number of WWW resources (HTML, image files, etc.) found within the proxy cache (i.e., 220). Parameter C denotes the percentage of the original cache that was relocated. Note that (2) assumes that P_{hit} is proportional to the total number of references to the items in the partial cache. Figure 6 plots P_{hit} as a function of C.

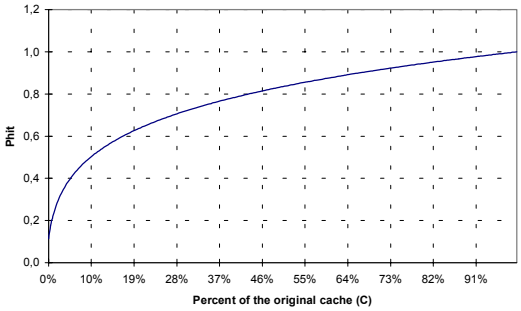

Figure 6. P_{hit} Vs relocation percentage

Figure 6 shows that a relocation of the most popular 70% of the original cache accomplishes a P_{hit} of 90%, while a relocation of the 30% gives a P_{hit} close to 70%. In our model, we have adopted these two values for the relocation percentages to the two, second best selections of the prediction algorithm and the three remaining neighboring cells, respectively. Figure 7 shows this approach.

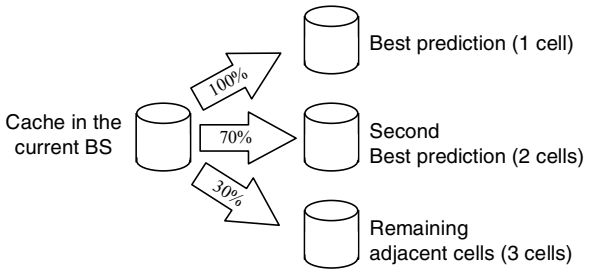

Figure 7. Cache relocation percentages

Let P_{new} denote the unconditional probability of a hit in the partial cache. We have

$$P_{new} = P_{old} \cdot P_{hit} \qquad (3)$$

where P_{old} is the probability of a hit in the original cache. Here we assume that P_{old}=0.2, consistently with our measured hit rate of 20%.

P_{new} is a measure of the efficiency of the partial cache right after its relocation in a neighboring BS. In the event of misses (i.e., the designated item could not be found in the partial cache), the numerator in (2) changes and P_{new} needs to be updated. The

denominator in (2) remains unchanged as it is only dependent on the total allowable set of cached items (i.e., the cache size) which, in our case, is kept constant. If, at some point t in time, a reference to a resource, Req(t), caused a cache miss (i.e., the requested resource was not found in the $\lfloor cache\_size \cdot C \rfloor + k$ items of the cache, where k is the number of resources previously fetched in the proxy cache as a result of other misses), or a cache hit, P_{new} is updated as in (4).

$$P_{new} = \begin{cases} P_{old} \cdot \dfrac{\sum_{i=1}^{\lfloor cache\_size \cdot C \rfloor + k} rank(i)^{-z} + (k+1)^{-z}}{\sum_{i=1}^{cache\_size} rank(i)^{-z}}, & \text{if Req(t) is a miss} \\ \text{unchanged}, & \text{if Req(t) is a hit} \end{cases} \quad (4)$$

Equation (4) shows, as expected, that P_{new} increases to P_{old}, as the proxy cache in the new BS fills up.

4. PATH PREDICTION ALGORITHM

The use of a path prediction algorithm in a mobile/wireless network architecture allows the efficient use of limited network resources such as disk capacity or bandwidth in BSs. A number of path prediction algorithms have been recently proposed in the literature of wireless data networks. Notable examples of such work are the algorithm proposed by Liu et al. [32] and the Liu-Maguire algorithm [29], [30]. The former example 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 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. 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.

The Liu-Maguire algorithm is based on Mobile Motion Prediction scheme for the prediction of the future location of a roaming user according to his movement history patterns. MMP is "based on the fact that everyone has some degree of regularity in his/her movement, that is, the movement of people consists of random movement and regular movement and the majority of mobile users has some regular daily (hourly, weekly, ...) movement patterns and follow these patterns more or less every day ...". The scheme consists of Regularity-Pattern Detection algorithms and Motion Prediction Algorithm. Regularity Detection is used to detect specific patterns of movement from a properly structured database (Itinerary Pattern Base). Three classes of matching schemes are used for the detection of patterns namely the state-, the velocity- and the frequency-matching. The Prediction Algorithm is invoked for combining regularity information with stochastic information (and constitutional constraints) and thus, reach a decision for the future location of the terminal. Simulations of the proposed scheme have shown a maximum prediction efficiency of 95%.

In this paper, we do not adopt any of the path prediction algorithms discussed above. Instead we propose a new algorithm, which is based on well-established Artificial Intelligence (AI) techniques for machine learning. More specifically, we adopt the use of a learning automaton [36]. In the past, there were also other proposals to consolidate AI techniques (e.g., genetic algorithms) with the technology of mobile communications for predicting terminal movement [31].

Learning automata are finite state adaptive systems that interact continuously in an iterative fashion with a general environment. Through a probabilistic, trial-and-error response process, they learn to choose or adapt to the behavior that generates the best response. In the first step of the learning process, an input is provided to the automaton from the environment. This input triggers one of a finite number of candidate responses from the automaton. The environment receives and evaluates the response and then provides feedback to the automaton. Such feedback is used by the automaton to alter its stimulus-response mapping structure to improve its behavior. Learning automata are generally considered as robust but not very efficient learners. They are relatively easy to implement. Generally, the operation of the learning automaton is based on a state transition matrix, which contains the one-step transition probabilities P_{ij} from the current state i to the next state j. Different approaches have been proposed for the updating of the state transition matrix after the reception of environment feedback. In this paper, we adopt the behavior of a Linear Reward-Penalty (L_{R-P}) scheme. When the automaton selects the right response, the positive feedback received by the environment causes the respective state transition to be "rewarded" (i.e., its probability is increased by some pre-arranged value) while the probabilities of the state transitions that were not selected (remaining transitions from the same state) are "penalized" (decreased) uniformly to keep the probability sum to 1. If the proposed response is not appropriate (i.e., a negative feedback was received from environment) a reverse approach is followed: the probability value of the selected transition is "penalized", while the remaining transitions are evenly rewarded to balance the decrease. This behavior is shown in (5).

$$\begin{aligned} &\text{Transition } (i \to j) \text{ received positive feedback: } \begin{cases} P_{ij} = P_{ij} + A \cdot (1 - P_{ij}) \\ P_{ik} = P_{ik} \cdot (1 - A),\ k \neq j \end{cases} \\ &\text{Transition } (i \to j) \text{ received negative feedback: } \begin{cases} P_{ij} = P_{ij} - A \cdot (1 - P_{ij}) \\ P_{ik} = P_{ik} \cdot (1 + A),\ k \neq j \end{cases} \end{aligned} \quad (5)$$

Upon invocation, the automaton selects, as candidate future state, the state with the highest probability. After consecutive interactions with the environment, some state transitions will have probabilities close to 1 while others will have near-zero values (automaton convergence). Our prediction algorithm capitalizes on the time - space regularity of the nomadic user's movement. To accomplish that, the entries of the transition matrix have the layout shown in Figure 8.

Previous Cell_ID	Current Cell_ID	Future Cell_ID	Time Slot	Probability Value	Time Stamp

Figure 8. Layout of state transition matrix entries

PreviousCell_ID, *CurrentCell_ID* and *FutureCell_ID* are the identifiers of the previous, current and possible future cell of the mobile terminal, respectively. *ProbabilityValue* denotes the likelihood that the mobile terminal (previously located at *PreviousCell_ID*) migrates, within a specific time slot (denoted by *TimeSlot*), from *CurrentCell_ID* to *FutureCell_ID*. Hence, each

state in the suggested algorithm is a triplet consisting of *PreviousCell_ID*, *CurrentCell_ID* and *TimeSlot*.

In the considered architecture, time is assumed divided in slots of 15 minutes (e.g., only values like 10:00, 10:15, 10:30, 10:45 can be found). Whenever a prediction request arrives at the home registry of the mobile terminal the entries pertaining to the chronologically closest time slot are taken into account provided that the distance to them does not exceed the 1 hour limit (i.e., 4 time slots). If no appropriate state transitions were found then new entries are introduced in the state transition matrix and a random one is chosen as algorithm's output.

The random selection described before is only influenced by the cell previously visited by the roaming terminal. In the absence of other information, we assume a linear movement of the terminal. Hence, the automaton, instead of making a completely random selection, points to the symmetrical cell to the one previously visited as the candidate cell.

If the automaton decision is correct (positive feedback) then the matrix entry that contributed to this decision is rewarded while the probability values of the remaining entries are reduced (penalized) according to (5). The reward/penalty procedure applies to all the entries of the matrix that refer to the same tuple of time slot, previous and current cell. When a mobile terminal powers-up in a new cell (not previously visited by the same terminal) a number of entries are automatically inserted in the matrix (also referred to as database or knowledge base). These entries contain equal probability values (i.e., 1/6 for hexagonal cells), the same current cell identification (*CurrentCell_ID*) and refer to the same time slot. The identification numbers of future (*FutureCell_ID*) and previous (*PreviousCell_ID*) cells reflect all the adjacent cells.

The *TimeStamp* field indicates when the particular entry was consulted for the last time. When the state space exceeds a pre-configured storage allotment, a garbage collection (GC) procedure is initiated and the oldest entries are deleted from the matrix[1] (i.e., a Least Recently Used, LRU, approach). This feature, in addition to time partitioning, prevents state space explosion.

5. SIMULATION MODEL: WWW TRAFFIC & CELL RESIDENCE TIME

To evaluate the effectiveness of the suggested cache relocation technique we simulated the movement of a nomadic WWW user throughout a wireless CPN. Below we discuss two very important issues for the simulation model, namely the modeling of the traffic generated by the WWW user as well as cell residence times.

5.1 WWW traffic modeling

In our model, the behavior of the nomadic WWW user was compliant with the traffic models and statistics reported within the WWW research community over the past years. Specifically, the random variable X used to model the transfer time for Web resources (documents, images, etc.) follows the Pareto (also known as power law) distribution [14], [15]:

$$F(x) = P[X \le x] = \begin{cases} 1-(\frac{v}{x})^{w}, & v \le x \\ 0, & x < v \end{cases} \qquad (6)$$

In (6), the shape parameter w has been estimated in the range 1.0-1.3 [14]. For the simulation, we have adopted w=1.2. As lower bound, v, for the random variable X we have used the value of 1.

In [14], it is shown that WWW traffic exhibits characteristics consistent with self-similar traffic models. Self-similarity is attributed to the multiplexing of a large number of ON/OFF sources where both the ON and the OFF period lengths are heavy tailed processes. ON times correspond to WWW resource transmissions while the OFF times correspond to intervals of browser inactivity. Furthermore, OFF times are classified either as Active (attributed to client processing delays: document parsing, resource rendering) or as Inactive (user think time). Active OFF times are in the range of 1 msec - 1 sec. Since the time granularity in our simulation environment is in the order of 1 sec, we model the client processing delay as a 1 sec constant. In [9], the Weibull distribution is used to model the Active OFF times. Inactive OFF times are modeled through the Pareto distribution with w=1.5 and v=1, [9].

A pictorial representation of the above assumptions is provided in Figure 9. As shown in Figure 9, the times needed for the transfer of additional resources (e.g., GIF images, etc.), the need for which is identified after parsing, can be overlapped (this is the approach followed by most contemporary multi-threaded WWW clients). In [9], the number of embedded references per document is calculated by applying thresholds to the Weibull distributed Active OFF times (i.e., if the Weibull distributed OFF time is less than the threshold value then it is assumed that an embedded reference has been encountered and the retrieval of the appropriate resource has been initiated). In the same paper, the distribution of the count of embedded references has been shown to follow the Pareto distribution (with parameters v=1 and w=2.43). Based on the assumption that the transmission of embedded references can be overlapped (Figure 9), additional resource transfer is invoked as a result of a Bernoulli trial with parameter q. The Bernoulli parameter q can be calculated from the Weibull distribution as the probability of exceeding the 1 sec threshold of Active OFF times (this event can be interpreted as the absence of embedded references in the document):

$$q = P(\text{OFF time} > 1) = 1-(1-e^{-(\frac{1}{a})^{b}}) = e^{-(\frac{1}{a})^{b}} \qquad (7)$$

Equation (7), for a=1.46 and b=0.382 (values estimated in [9]) yields q=0.421. This figure is very close to the WWW statistics reported in [12]. On the basis of the above, the ON/OFF behavior of a WWW traffic source can be modeled through the chain shown in Figure 10.

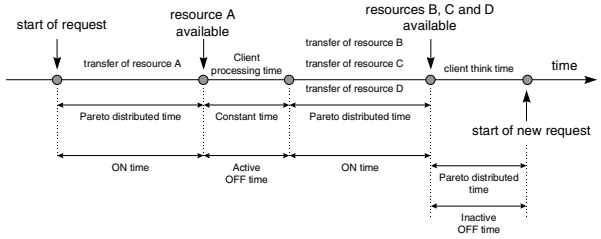

Figure 9. Time distributions for WWW browsing

[1] Provision is taken to keep the probability values consistent after the deletion of some entries.

Lastly, we should point out that the resource transmission times are not, somehow, correlated with resource sizes (i.e., file sizes). A large file may be retrieved from a site that is speedily accessed, while a small file may be fetched by a remote location with considerable time delays.

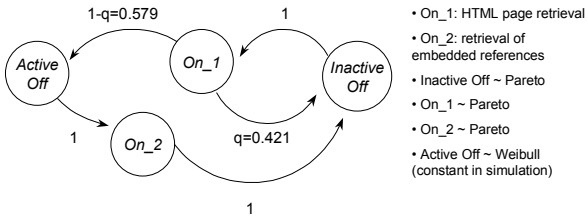

Figure 10. WWW traffic chain model

5.2 Cell residence times

Another key issue that we had to handle for the simulation environment was the modeling of the time spent by the nomadic user within the current cell. This time is referred to as cell residence time and is modeled by a random variable R. In modeling R, we considered two principal categories of nomadic users; those traversing the cell very rapidly (i.e., on a vehicle) and those crossing cell boundaries at a significantly lower rate (i.e., walking velocity). In view of the above, we assume that the probability distribution $F_R(\cdot)$ of R is as given below:

$$F_R(r) = f \cdot G_f(r) + (1-f)G_s(r) \qquad (8)$$

In (8), f denotes the fraction of the fast moving users over the entire nomadic user population ($0 \le f \le 1$); $G_f(\cdot)$ and $G_s(\cdot)$ denote the probability distributions of a fast and a slow moving user, respectively. Both $G_f(\cdot)$ and $G_s(\cdot)$ are assumed truncated Normal distributions to avoid negative and unrealistically small residence times and are as given below.

$$G_f(r) = \begin{cases} \dfrac{N_f(r) - N_f(t_f)}{1 - N_f(t_f)}, & t_f < r < +\infty \\ 0, & -\infty < r \le t_f \end{cases}$$

$$G_s(r) = \begin{cases} \dfrac{N_s(r) - N_s(t_s)}{1 - N_s(t_s)}, & t_s < r < +\infty \\ 0, & -\infty < r \le t_s \end{cases}$$

where $N_f(\cdot)$, $N_s(\cdot)$ are normal distributions, and t_f, t_s are non-negative threshold parameters. In our simulation, we have used $t_f = t_s = 3$ sec, $f = 0.3$, and means and standard deviations for $N_f(\cdot)$ and $N_s(\cdot)$ as shown in Table 1.

6. SIMULATION RESULTS

6.1 Simulation of the path prediction algorithm

We have programmed the logic of the automaton-based path prediction algorithm in Prolog, a 5th generation (declarative) programming language widely used in AI applications [11]. Specifically, we have made use of the Arity/Prolog interpreter [6].

Table 1. Basic simulation parameters

Parameter	Value
A) Efficiency of the path prediction algorithm	
• Probability for Perfect Hit	0.48
• Probability for Hit to Second Best predictions	0.20
B) Cell residence time	
• mean of $N_f(\cdot)$	45 sec
• standard deviation of $N_f(\cdot)$	20 sec
• mean of $\boldsymbol{N_s(\cdot)}$	160 sec
• standard deviation of $N_s(\cdot)$	30 sec
• percentage f (fast moving users / total population of users)	30%
• cut-off threshold (truncation point)	3 sec
C) Web browsing	
• Pareto distribution for resource transmission times: w shape parameter	1.2
• Pareto distribution for resource transmission times: v lower bound	1
• probability for embedded references, q (see Figure 10)	0.579
• Pareto distribution for inactive OFF times: w shape parameter	1.5
• Pareto distribution for inactive OFF times: v lower bound	1
D) Proxy cache performance	
• Number of items (2MB cache per terminal)	220
• Hit rate (2MB cache)	20%
• Relocation percentage for best prediction	100%
• Relocation percentage for second best predictions (2 cells)	70%
• Relocation percentage for other neighboring cells (3 cells)	30%

We have logged the behavior of the automaton throughout a period of 7 days for a total of 592 handovers. The automaton was applied to the movement of one of the authors within the building of our Department. The movement patterns which were fed to the automaton were quite similar in terms of cell border crossings (i.e., movement from room X to room Y always followed the same route). The movement patterns were fed to the automaton with considerable time variance (i.e., similar itineraries were fed to the automaton in various - not exactly random - time instances of the considered days).

As shown in Figure 11, which plots the performance achieved by the automaton, low hit rates were logged during the first 3 days of the algorithm execution. During those days, due to the absence of state information, the automaton constantly populated the transition matrix (see Figure 12). Since no relevant entries were found in the knowledge base, the algorithm's successful predictions logged during those days were mainly attributed to the assumed linear movement of the user. In the absence of other information, the cell previously visited by the roaming terminal influences future cell prediction. The automaton, instead of making a random selection, points as candidate cell the symmetrical cell to the one previously visited. This strategy is not applied if some of the involved probabilities is greater than 1/6.

In Figure 11, we have plotted the probability that the main algorithm output is the one to which the terminal has really been handed over ("Perfect Hit" series/line). We have also plotted the probability that

some cell, classified by the automaton as the second best prediction (to the one which has been selected), is the cell to which the terminal has actually been handed over (such probability is reflected by the "Hit to Second Bests" series/line). We have performed this classification for two additional cells (i.e., the automaton output consisted of the three most probable future cells). The logging of this probability was necessary since the proposed algorithm is largely based on the partial relocation of the accumulated cache to certain neighboring cells (i.e., 70% to the "Second Best" predictions). In principle, we show that a relatively simple prediction algorithm can achieve quite similar performance with the more complex schemes [7], if combined with an approach like the Shadow Cluster.

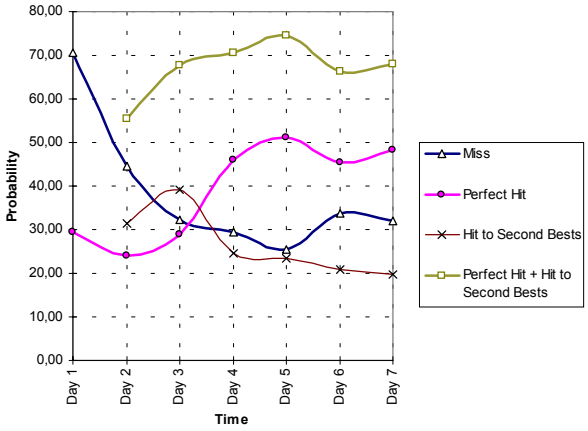

Figure 11: Path prediction efficiency

6.2 Simulation of the cache relocation scheme

The simulation model for the cache relocation scheme was programmed in MS-Visual C++ ver.5. The most important metrics monitored through the simulation program were

- the average delay (waiting time) perceived by the user in his WWW requests,

- the number of WWW connections that were interrupted by handovers,

- the percentage of interrupted WWW connections (i.e., [handover interrupted connections] / [total number of connections]),

- the number of cache items which were relocated from the current BS to the BS used by the MT after the occurrence of a handover, and,

- the achieved cache hit rate.

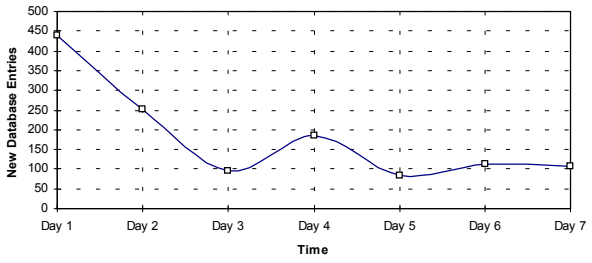

Figure 12: New entries in the automaton matrix

The basic parameters adopted for the simulation program are summarized in Table 1.

We have simulated three scenarios. In **Scenario 1**, the cache relocation scheme described above was employed in the wireless infrastructure. In **Scenario 2**, proxy caches existed in BSs but were not relocated; no prediction algorithm was used. In **Scenario 3**, no proxy caching was applied in the various BSs (hence, all requested resources were fetched from remote servers). For each scenario we performed 21 trials involving 300 handovers each. In Figure 13 we plot the average waiting time (or delay time) per request for all three scenarios. Scenario 1 achieves a 77.8% of the time consumed in Scenario 3, while Scenario 2 achieves a 96.3 %. In terms of handover interrupted connections, the achieved mean percentage values are presented in Table 2.

Table 2. Mean percentage of HO interrupted connections

Scenario 1	Scenario 2	Scenario 3
2.51 %	2.78 %	3.02 %

The observations from Table 2 are crucial since the HTTP uses TCP as a reliable transport protocol. Apart from the well-known problems in the interaction of the stateless HTTP (ver.1.0) with TCP [37], Caceres and Iftode in [13] have shown how TCP's congestion control mechanism undermines communication throughput during handovers. The above researchers have quantified the performance degradation in TCP connections caused by MT movement across cell-boundaries. In an overlapping cell scenario, handovers cause the throughput of the TCP connection to decrease by 6%, while in the non-overlapping cell scenario with 0-sec rendezvous delay[2] throughput drops by 12%. Owing to the stateless character of the HTTP, TCP connections conveying HTTP messages are generally short-lived (their duration equals the time of the WWW resource transmission time discussed earlier in this paper). Limiting the probability of interruption, by a handover, of a WWW connection will, thus, be beneficial for the performance of the underlying TCP layer. Apart from Average Waiting Time and the Percentage of handover interrupted connections, we have also logged, in Scenario 1, the number of items that were relocated in each handover occurrence. In the 14.3% of handovers the relocated volume of data is lower than that allowed by the cache relocation scheme. For example, although the algorithm allows the relocation of 154 items to second best predictions, a lesser number of items existed in the original cache and were eventually relocated. Thus, the utilization factor of the relocation scheme dropped. This utilization factor is increased only if the normally distributed cell residence times are systematically broadened. In such case, the cache relocation scheme has the opportunity of recovering from prediction misses since the local cache gradually fills up to the maximum allowed size even though only a percentage of the original set was received from the previous BS. We should also mention that, in another series of experiments where, instead of the suggested distribution of cell residence times, a simpler exponential model was adopted, the percentage of non-optimum handover relocations was close to 17.5 %. In terms of proxy cache hit rates, Scenario 1 achieves a 16.76% while Scenario 2 achieves only a 7.78% (less than half of the rate accomplished in Scenario 1).

[2] The MT receives control information from the adjacent cell as soon as it leaves the current cell.

7. Conclusions

In this paper, we have proposed a scheme for improving the operation of the WWW in cellular CPNs. We have briefly discussed some of the relevant architectures which have been recently proposed for the introduction of the WWW in wireless networks. Most of these architectures target the optimization of the HTTP communication by means of proxies installed in the access network. Provision is not made, though, for the relocation of the proxy cache when the mobile terminal moves to different BS as a result of a handover. We have proposed the pro-active relocation of the proxy cache on the basis of the outcome of a path prediction algorithm. The proposed algorithm, which is based on the well-established learning automaton AI technique, takes into account the time-space regularity in the movement of nomadic users. The algorithm is executed at the access sub-network where the mobile terminal is registered and its implementation is fairly simple. Cache relocation is performed to all the cells adjacent to the one that is currently used. The percentages of the original cache that are relocated are affected by a categorization of cells provided by the prediction algorithm. We have simulated the prediction algorithm that demonstrated a hit rate of approximately 48%. The cache relocation scheme may also benefit from the second best guesses of the algorithm, which seem to successfully match the real route of the mobile terminal with a probability of 0.2. Based on such figures we proceeded with the simulation of the overall scheme. For its simulation we have taken into account the self-similar nature of WWW traffic. The simulation of the proposed algorithm revealed a decrease in the order of 22.2% in the delay times experienced by the nomadic users. Additionally, the rapid dispatch of WWW queries due to the relocated cache proves beneficial for the TCP layer.

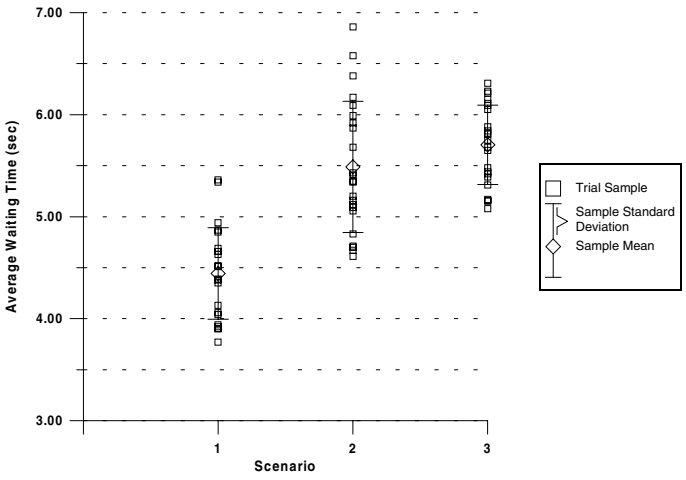

Figure 13. Average waiting time for the different scenarios

Our work is well aligned with the research directions in the WAP Forum - W3C co-operation for bandwidth efficiency through the use of "Smart Web Proxies" [24]. A potential improvement of the current work would be the study of more robust learners for use in the path prediction algorithm. Specifically, we are considering the promising possibility of employing Fuzzy Set theory for ameliorating the decisions taken by the algorithm.

8. REFERENCES

[1] M. Abrams, et al., "Caching Proxies: Limitations and Potentials," Proceedings of 4th International WWW Conference, Boston - MA, USA, December 1995.

[2] A. Acharya, et al., "Mobility Management in Wireless ATM Networks," IEEE Communications Magazine, Vol.35, No.11, November 1997.

[3] C. Aggraval, et al., "On Caching Policies for Web Objects," IBM Research Report RC20619, May 1997.

[4] B. Akyol and D. Cox, "Signaling Alternatives in a Wireless ATM Network," IEEE JSAC, Vol.15, No.1, January 1997.

[5] V. Almeida, et al., "Characterizing Reference Locality in the WWW," Technical report BU-CS-96-11, Computer Science Dept., Boston University, 1996.

[6] "The Arity/Prolog Compiler and Interpreter," Arity Corporation, Concord, MA, 1992.

[7] P. Bahl and V. Padmanabhan, "User Location and Tracking in an In-Building Radio Network," Microsoft Research, Technical Report MSR-TR-99-12, Feb 1999.

[8] C. Baquero, et al., "MobiScape: WWW Browsing under Disconnected and Semi-Connected Operation," Proceedings of 1st Portuguese WWW National Conference, Braga, Portugal, July 1995.

[9] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," Proceedings of ACM SIGMETRICS, July 1998.

[10] T. Berners-Lee, et al., "The World-Wide Web," CACM, Vol.37, No.8, August 1994.

[11] I. Bratko, "Prolog Programming for Artificial Intelligence," Addison-Wesley, 1990.

[12] T. Bray, "Measuring the Web," Computer Networks and ISDN Systems, Vol. 28, No. 7-11, 1996.

[13] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," IEEE JSAC, Vol.13, No.5, June 1995.

[14] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," IEEE/ACM Transactions on Networking, Vol. 5, No. 6, December 1997.

[15] M. Crovella, et al., "Heavy-Tailed Probability Distributions in the World Wide Web," in "A Practical Guide to Heavy Tails - Statistical Techniques and Applications", R. Adler, R. Feldman, and M. Taqqu (ed.), BIRKHAUSER, 1998.

[16] C. Cunha, et al., "Characteristics of WWW Client-based Traces," Technical report BU-CS-95-010, Computer Science Dept., Boston University, July 1995.

[17] K.Y. Eng, et al., "A Wireless Broadband Ad-Hoc ATM Local-Area Network," ACM/Baltzer Wireless Networks Journal, Vol. 1, May 1995.

[18] "Radio Equipment and Systems (RES); Digital European Cordless Telecommunications (DECT) - Common Interface - Part 1: Overview," European Telecommunication Standard (ETS 300 175-1), ETSI, October 1992.

[19] "Universal Mobile Telecommunication System (UMTS), Service Aspects, Virtual Home Environment, v.2.0.0," UMTS 22.70, ETSI, March 1998.

[20] R. Floyd, et al., "Mobile Web Access Using eNetwork Web Express," IEEE Personal Communications, October 1998.

[21] S. Glassman, "A Caching Relay for the World Wide Web," Computer Networks and ISDN Systems, Vol.27, No.2, 1994.

[22] S. Hadjiefthymiades and L. Merakos, "Improving the Performance of the World Wide Web in Cellular CPN Environments," Proceedings of MoMuc'98, Berlin, Germany, October 1998.

[23] G.A. Halls, "HIPERLAN: the high performance radio local area network standard," Electronics and Communication Engineering Journal, December 1994.

[24] J. Hjelm, et al., "WAP Forum - W3C Cooperation White Paper," Sept. 1998.

[25] B.C. Housel and D.B. Lindquist, "WebExpress: A system for Optimising Web Browsing in a Wireless Environment," Proceedings of ACM/IEEE MobiCom '96, New York, USA, October 1996.

[26] ITU-T Recommendation Z.120, "Message Sequence Chart (MSC)," 1993.

[27] T.F. La Porta, et al., "Challenges for Nomadic Computing: Mobility Management and Wireless Communications," ACM Journal of Nomadic Computing, Vol.1 No.1, 1996.

[28] D. Levine, et al., "A Resource Estimation and Call Admission Algorithm for Wireless Multimedia Networks Using the Shadow Cluster Concept," IEEE/ACM Transactions on Networking, Vol.5, No.1, February 1997.

[29] G.Y. Liu and G.Q. Maguire, Jr., "A Predictive Mobility Management Algorithm for Wireless Mobile Computing and Communications," Proceedings of ICUPC '95, Tokyo, Japan, November 1995.

[30] G.Y. Liu and G.Q. Maguire, Jr., "Efficient Mobility Management for Wireless Data Services," Proceedings of IEEE VTC '95, Chicago - Illinois, USA, 1995.

[31] L.Q. Liu, et al., "Efficient Mobility Management: A New Flexible Design Algorithm," Proceedings of ICUPC '96, Cambridge - MA, USA, Sept. 1996.

[32] T. Liu, et al., "Mobility Modeling, Location Tracking, and Trajectory Prediction in Wireless ATM Networks," IEEE JSAC, Vol.16, No 6, August 1998.

[33] A. Luotonen and K. Altis, "World-Wide Web Proxies," Proceedings of 1st International WWW Conference, Geneva, Switzerland, May 1994.

[34] J. Mikkonen, et al., "The Magic WAND - Functional Overview," IEEE JSAC, Vol.16, No.6, August 1998.

[35] M. Mouly and M. Pantet, "The GSM System for Mobile Communications," ISBN: 2-9507190-0-7

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

[37] V.N. Padmanabhan and J.C. Mogul, "Improving HTTP Latency," Computer Networks and ISDN Systems Vol.28, 1995.

[38] "The Path towards UMTS - Technologies for the Information Society," UMTS Forum Report, 1998.

[39] M. Viktora and M. Rubáš, "WinProxy 1.3, User's Manual," LAN-Projekt, October, 1996.