Location Prediction Algorithms for Mobile Wireless Systems

Christine Cheng
Dept. of EECS
Univ. of Wisconsin - Milwaukee
3200 N. Cramer St.
Milwaukee, WI 53211
ccheng@cs.uwm.edu

Ravi Jain
DoCoMo USA Labs
181 Metro Dr
San Jose, CA 95110
jain@docomolabs-usa.com

Eric van den Berg
Applied Research
Telcordia Technologies
445 South St
Morristown, NJ 07960
evdb@research.telcordia.com

September 10, 2002

Abstract

Predicting the location of a mobile wireless user is an inherently interesting and challenging problem. Location prediction has received increasing interest over the past decade, driven by applications in location management, call admission control, smooth handoffs, and resource reservation for improved Quality of Service. It is likely that location prediction will receive even more interest in the future, especially given the increased availability and importance of location estimation hardware and applications.

In this chapter we present an overview of location prediction in mobile wireless systems. We do not attempt to provide a comprehensive survey of all techniques and applications, but instead a description of several types of algorithms used for location prediction. We classify them broadly into two types of approaches: domain-independent algorithms that take results from Markov analysis or text compression algorithms and apply them to prediction, and domain-specific algorithms that consider the geometry of user motion as well as the semantics of the symbols in the user's movement history. We briefly also mention other algorithms using Bayesian or neural network approaches, and end with some concluding remarks.

1 Introduction

Predicting the location of a user or a user's mobile device is an inherently interesting problem and one that presents many open research challenges. The explosion in mobile wireless

technologies and applications over the past decade has sparked renewed interest in location prediction techniques. The advent of new access technologies such as wireless Local Area Network (LAN) and Third Generation (3G) systems, location-based services, and pervasive computing and communications indicate that location prediction will become even more important in the future.

There are two classes of applications that can benefit from accurate prediction of a user's location: end-user applications, where the object is to predict location so that a human user can prepare or react accordingly, and system-enhancement applications, where the location prediction can be used to enhance system performance, availability or other metrics. An example of an end-user application is one that predicts the location of a moving vehicle for road traffic optimization or for catching thieves if the vehicle is stolen. An example of a system-enhancement application is to predict the location of a moving vehicle where a passenger is using a cell phone so as to reserve resources in adjoining cells and hence provide a smooth handoff.

Location can be specified in an absolute coordinate system e.g. latitude/longitude, or in symbolic coordinates (e.g. cell ID). In some cases both may be available. For example a facilities administrator in an office building is likely to have a detailed map of the room layout showing both absolute locations (in meters from some fixed origin) as well as symbolic locations (room numbers).

While in principle the same basic prediction techniques can be used for both end-user and system-enhancement applications, the constraints and metrics differ. For example, in end-user applications it may be important to know the user's geographical location, while for a system-enhancement application knowing parameters required for signaling (e.g. cell ID or paging area) is more relevant. In this chapter we have assumed that system-enhancement applications are the target. We assume that time is discretized and a user's location is given in symbolic coordinates. The task of the location prediction algorithm is to provide the user's (symbolic) location at the next time step, or, if possible, the path of the user (a sequence of locations) over several time steps. Note that the user's predicted location at the next time step may be the same as the current location.

Location prediction has been implicitly or explicitly utilized in many areas of mobile and wireless system design. For example, consider the problem of locating a cellular phone user in order to deliver a call to that user, or more specifically, determining which set of cells, called a location area (or registration area or paging area), the user is currently located in. Broadly speaking, the strategies employed in cellular systems essentially consist of having the mobile device report its location area to a set of databases which are queried when an incoming call arrives for the user [17, 2]. Analysis showed that these strategies placed a heavy burden on the SS7 signaling network used in the PSTN wired backbone, in particular on the Home Location Register (HLR) database in the user's home network. Early work on reducing this signaling impact used the following simple idea: the caller's switch recorded (cached) the location area at which the called party was found when the switch last queried the HLR database [16]. For the new call it attempted to locate the user at that location area first, and only queried the HLR if the user was no longer found there. Thus essentially the switch using this caching strategy employed a simple location prediction algorithm in order to reduce the overall signalling load in the system. As we discuss later in this chapter, this is a type of order-1 Markov predictor where the next term in the sequence is assumed to be identical to the present term. In this example, as in other applications, in abstract terms the location prediction algorithm is worthwhile if, over the entire population of users:

$$pS > A + (1-p)F \quad (1)$$

where p is the probability of successful prediction, S is the benefit of success, A is the cost of running the prediction algorithm itself, and F is the cost of failure. Of course, this general relation has to be made specific and evaluated for any particular application, architecture, and prediction algorithm.

We briefly mention types of location and mobility prediction that we do not consider in this chapter. Efforts on location management and prediction for other types of mobile objects, including software objects such as agents [36, 28], are outside the scope of the chapter. We also do not cover efforts on predicting the amount of time that the wireless link that a mobile host is using will stay usable (e.g. [34]), or on predicting the link quality and availability [18].

The rest of this chapter is organized as follows. In the following section we begin with some definitions and preliminaries. In sec. 3 we describe location prediction algorithms that do not explicitly take advantage of the specifics of the mobile wireless environment. These algorithms generally are based on order-k Markov prediction or on the prediction capabilities inherent in text compression algorithms. In sec. 4 we describe algorithms that have been designed for location prediction in mobile wireless environments and do explicitly take advantage of their characteristics. Location prediction techniques have been developed or suggested for many domain-specific applications including location management (e.g. [19, 33, 23] and references therein), smooth handoffs (e.g. [22, 23, 9, 14] and references therein), resource reservations (e.g. [21, 5, 31, 27, 12, 37] and references therein), call admission control (e.g. [29, 30, 26, 11, 24, 38] and references therein) and adaptive resource management (e.g. [22, 6] and references therein). We do not attempt to provide a comprehensive survey of all these domain-specific techniques; instead we briefly present some domain-specific algorithms that suggest slightly different approaches to the prediction problem.

All the algorithms we discuss essentially compare the sequence of recent movements the user has made to the sequence of locations $\mathcal{H}$ representing users' (or this particular user's) stored movement history. One way that the domain-independent algorithms discussed in sec. 3 differ from the domain-specific algorithms discussed in sec. 4 is in how the stored history $\mathcal{H}$ is partitioned into substrings for the purposes of this comparison. The order-k Markov predictors (sec. 3.1) essentially compare the most recent k movements of the user with every length k substring in $\mathcal{H}$. The LZ-based predictors (sec. 3.2) partition $\mathcal{H}$ based on techniques used in text compression algorithms. In contrast, the domain-specific algorithms partition the history based on the semantics of the location prediction domain, such as considering a location to be a substring delimiter if the user was stationary there a significant amount of time or if the location is at the boundary of the geographical service area.

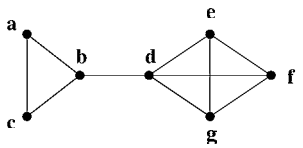
2 Preliminaries

2.1 Movement history

We will assume that the user's location is given in symbolic coordinates, and that the system has a record of the user's past movements based on its location updates. The user's *movement history* is thus represented as a sequence of symbols from an alphabet $\mathcal{A}$ which is finite. The information contained in the record depends on the system's update scheme. How this

information is interpreted also affects the results of prediction algorithms.

For example, in movement-based update schemes, updates occur every time the user has crossed M cell boundaries [1, 3]. If $M = 1$ then a record can look like the table below which has the details of all the user's cell crossings of the map on the left from 9 AM to 10 AM.



Time	9:00	9:04	9:18	9:20	9:31	9:43	9:56
Crossing	a	b	d	e	f	d	b

Figure 1: Example of a cell boundary graph and movement history

In this case, each symbol of the sequence is an ordered pair (t, v) where t is the time of update (and is discretized) and v is the user's new location. Depending on the purpose of a prediction algorithm, this sequence may be transformed to a new one so that the size of $\mathcal{A}$ is smaller [1]. For instance, if a prediction algorithm's objective is to predict the user's next cell, it may just consider the sequence $abdefdb$. In this case, $\mathcal{A}' = V$, the set of all the cell ID's.

For a time-based update scheme such as [32], updates occur every T time units. If we set $T = 5$ minutes, then the sequence generated from Table 1 would be $abbbeeeffdddb$. Notice that while this sequence is able to capture the duration of residence of the user at a cell, it misses the cell crossings that took place between updates. It will, nonetheless, be useful for a prediction algorithm that seeks to predict the user's location in the next T time units.

In [7], the authors suggested generating a movement history that reflects both cell crossings and durations of residence of the user at each cell while keeping $\mathcal{A} = V$. Such a history is generated when a user updates every T time units and every M cell crossings. If $T = 5$ and $M = 1$, the sequence that reflects the movement in Table 1 would be $abbbbdeeeffffddddb$.

Hence, there are different ways of representing a user's movement history as a sequence from a finite alphabet. It is imperative that the choice of sequence be matched to the purpose(s) of the prediction algorithm.

In the following we will assume that the appropriate movement history has been chosen. A history of length n is denoted as a sequence $\mathcal{H}_n =< X_1 = a_1, \ldots, X_n = a_n >$, where each X_i is a random variable and each $a_i \in \mathcal{A}$. For brevity we will sometimes denote a sequence as $a_1 a_2 \ldots a_n$. The notation $P(X_i = a_i)$ denotes the probability that X_i takes the value a_i and $\widehat{P}(X_i)$ denotes an estimate of $P(X_i)$.

2.2 Approach

We will discuss different prediction algorithms which use different approaches to predict the next term of the user's itinerary, i.e., sequence L. When possible, we also discuss how these methods can be extended to predict not just the next term but the future terms of the sequence.

Prediction algorithms usually often of two steps: the first step is to assign conditional probabilities to the elements of $\mathcal{A}$ given the user's movement history $\mathcal{H}$; the second step is to use these values to predict the next term in the sequence.

[1] Intuitively, the smaller $|\mathcal{A}|$ is, the better because there will be fewer choices for a prediction.

We note that there are some applications where a single guess for the next term may be too restrictive. For example, to satisfy QoS requirements, Chan, et al in [10] proposed an algorithm that outputs a subset of $\mathcal{A}$ so that the probability that the next term of the movement history is in this set is above some threshold.

3 Domain-independent Algorithms

We discuss two families of domain-independent algorithms that have been used as the core of techniques for location prediction in mobile wireless systems.

3.1 The order-k Markov Predictor

The order-k Markov predictor assumes that the next term of the movement history depends only on the most recent k terms. Moreover, the next term is independent of time. That is, if the user's history consists of $\mathcal{H}_n = \{X_1 = a_1, \ldots, X_n = a_n\}$, then, for all $a \in \mathcal{A}$,

$$\begin{aligned} P(X_{n+1} = a|\mathcal{H}_n) &= P(X_{n+1} = a|X_{n-k+1} = a_{n-k+1}, \ldots, X_n = a_n) \\ &= P(X_{i+k+1} = a|X_{i+1} = a_{n-k+1}, \ldots, X_{i+k} = a_n), \quad \forall i \in \mathbb{N}. \end{aligned}$$

The *current state* of the predictor is assumed to be $< a_{n-k+1}, a_{n-k+2}, \ldots, a_n >$.

If the movement history was truly generated by an order-k Markov source then there would be a *transition probability matrix* M that encodes these probability values. Both the rows and columns of M are indexed by length-k strings from $\mathcal{A}^k$ so that $P(X_{n+1} = a|\mathcal{H}_n) = M(s, s')$ where s and s' are the strings $a_{n-k+1}a_{n-k+2}\ldots a_n$ and $a_{n-k+2}a_{n-k+3}\ldots a_n a$ respectively. In that case, knowing M would immediately provide the probability for each possible next term of $\mathcal{H}_n$.

Unfortunately, even if we assume the movement history *is* an order-k Markov chain for some k we do not know M. Here is how the order-k Markov predictor estimates the entries of M. Let $N(s, s')$ denote the number of times the substring s' occurs in the string s. Then, for each $a \in \mathcal{A}$,

$$\widehat{P}(X_{n+1} = a|\mathcal{H}_n) = \frac{N(a_{n-k+1}\ldots a_n a, \mathcal{H}_n)}{N(a_{n-k+1}\ldots a_n, \mathcal{H}_n)} \tag{2}$$

If r predictions are allowed for X_{n+1} then the predictor chooses the r symbols in $\mathcal{A}$ with the highest probability estimates. In other words, the predictor always chooses the r symbols which most frequently followed the string $a_{n-k+1}\ldots a_n$ in $\mathcal{H}_n$.

Vitter and Krishnan suggested using the above predictor in the context of prefetching web pages [35]. Chan et al [10] considered five prediction algorithms, three of which can be expressed as order-k predictors. (We will briefly describe the other two in a later section.) Two of them, the location-based and direction-based prediction algorithms are equivalent to the order-1 and order-2 Markov predictors respectively when $\mathcal{A}$ is the set of all cell id's. The time-based prediction algorithm is a order-2 Markov predictor when $\mathcal{A}$ is the set of all time-cell id pairs.

We emphasize that the above prediction scheme can be used even if the movement history is *not* generated by an order-k Markov source. If the assumption about the movement history is true, however, the predictor has the following nice property. Consider $\mathcal{F}$, the family of

prediction algorithms that make their decisions based only on the user's history, including those that have full knowledge of the matrix M. Suppose each predictor in $\mathcal{F}$ is used *sequentially* so that a guess is made for each X_i. We say that a predictor has made an error at step i if its guess $\widehat{X}_i$ does not equal X_i. Let $\widehat{\pi}(\mathcal{H}_n) = \sum_{i=1}^{n} I(\widehat{X}_i \neq X_i)/n$, where I is the indicator function, denote the average error rate for the order-k Markov predictor. Let $\pi_{\mathcal{F}}(\mathcal{H}_n)$ be the best possible average error rate achieved by any predictor in $\mathcal{F}$. Vitter and Krishnan [35] showed that as $n \to \infty, \widehat{\pi}(\mathcal{H}_n) \to \pi_{\mathcal{F}}(\mathcal{H}_n)$. That is, the average error rate of the order-k predictor is best possible as $n \to \infty$, or is *asymptotically optimal*. This result holds for any given value of k.

We observe that the algorithm can (naively) be generalized for predicting location beyond the next cell, i.e, predicting the user's path, as follows. If M is known, $P(X_t = a|\mathcal{H}_n)$ for any $t > n+1$ and each $a \in \mathcal{A}$ can be determined exactly from $M^{(t-n)}$. The process to estimate $M^{(t-n)}$ is to first construct $\widehat{M}$, the estimate for M, and then raising it to the $(t-n)$th power. Then the value(s) of X_t can be predicted using the same procedure as for X_{n+1}. However, any errors in the estimate of M will accumulate as prediction is attempted for further steps in the future.

3.2 The LZ-based Predictors

LZ-based predictors are based on a popular incremental parsing algorithm by Ziv and Lempel [39] used for text compression. Some of the reasons why this approach was considered were (a) most good text compressors are good predictors [35] and (b) LZ-based predictors are like the order-k Markov predictor except that k is a variable allowed to grow to infinity [7]. Let us first describe the Ziv-Lempel parsing algorithm.

3.2.1 The LZ parsing algorithm

Let γ be the empty string. Given an input string s, the LZ parsing algorithm partitions the string into distinct substrings $s_0, s_1, \ldots, s_m$ such that $s_0 = \gamma$, for all $j \geq 1$, substring s_j without its last character is equal to some $s_i, 0 \leq i < j$, and $s_0 s_1 \ldots s_m = s$. Observe that the partitioning is done sequentially, i.e., after determining each s_i the algorithm only considers the remainder of the input string. For example, $\mathcal{H}_n = abbbbdeeefffddddb$ is parsed as $\gamma, a, b, bb, bd, e, ee, f, ff, d, dd, db$.

Associated with the algorithm is a tree, which we call the *LZ tree*, that is grown dynamically to represent the substrings. The nodes of the tree represent the substrings where node s_i is an ancestor of node s_j if and only if s_i is a prefix of s_j. Typically, statistics are stored at each node to keep track of information such as the number of times the corresponding substring has been seen as a prefix of $s_0, s_1, \ldots s_m$ or the sequence of symbols that has followed the substring. The tree associated with the above example is shown in Fig. 2.

The process of determining substring s_i is equivalent to tracing a path starting from the root of the LZ tree until a leaf labelled with s_i is reached. A new node is then added as a child to the leaf and labelled with s_j. Path tracing then resumes at the root.

3.2.2 Applying LZ to prediction

Different predictors based on the LZ parsing algorithm have been suggested in the past [35, 20, 15, 7, 37]. We describe some of these below and then discuss how they differ.

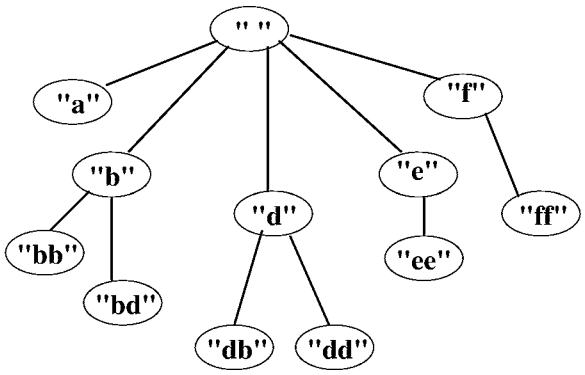


Figure 2: An example LZ parsing tree

Suppose $\mathcal{H}_n$ has been parsed into $s_0, s_1, \ldots, s_m$. If the node associated with s_m is a leaf of the LZ tree then LZ-based predictors usually assume that each element in $\mathcal{A}$ is equally likely to follow s_m. That is, $\widehat{P}(X_{n+1} = a|\mathcal{H}_n) = 1/|\mathcal{A}|$. Otherwise, LZ-based predictors estimate $P(X_{n+1} = a|\mathcal{H}_n)$ based on the symbols that have followed s_m in the past when $\mathcal{H}_n$ was parsed.

1. Vitter and Krishnan [35] considered the case when the generator of $\mathcal{H}_n$ is a finite-state Markov source, which produces sequences where the next symbol is dependent on only its *current state*. (We note that a finite-state Markov source is different from the order-k Markov source in that the states do not have to correspond to strings of a fixed length from $\mathcal{A}$.) They suggested using the following probability estimates: for each $a \in \mathcal{A}$, let

$$\widehat{P}(X_{n+1} = a|\mathcal{H}_n) = \frac{N^{LZ}(s_m a, \mathcal{H}_n)}{N^{LZ}(s_m, \mathcal{H}_n)} \tag{3}$$

   where $N^{LZ}(s', s)$ denotes the number of times s' occurs as a prefix among the substrings $s_0, \ldots, s_m$ which were obtained by parsing s using the LZ algorithm.

   It is worthwhile comparing Eq.(2) with Eq.(3). While the former considers how often the string of interest occurs in the entire input string, (i.e, in our application, the history $\mathcal{H}_n$) the latter considers how often it occurs in the partitions s_i created by LZ. Thus, in the example of Fig. 2, while *bbb* occurs in $\mathcal{H}_n$ it does not occur in any s_i.

   If r predictions are allowed for X_{n+1}, then the predictor chooses the r symbols in $\mathcal{A}$ that have the highest probability estimates. Once again, Vitter and Krishnan showed that this predictor's average error rate is asymptotically optimal when used sequentially.

2. Feder, Merhav and Gutman [15] designed a predictor for arbitrary binary sequences, i.e., sequences where $\mathcal{A} = \{0, 1\}$. The following are their probability estimates:

$$\begin{aligned} \widehat{P}(X_{n+1} = 0|\mathcal{H}_n) &= \frac{(N^{LZ}(s_m 0) + 1)}{(N^{LZ}(s_m 0) + N^{LZ}(s_m 1) + 2)} \\ &= 1 - \widehat{P}(X_{n+1} = 1|\mathcal{H}_n). \end{aligned} \tag{4}$$

   For convenience let $\alpha = \widehat{P}(X_{n+1} = 0|\mathcal{H}_n)$. Then the predictor guesses that the next term is 0 with probability $\phi(\alpha)$ where for some chosen $\epsilon > 0$

$$\phi(\alpha) = \begin{cases} 0 & 0 \le \alpha < \frac{1}{2} - \epsilon \\ \frac{1}{2\epsilon}\left[\alpha - \frac{1}{2}\right] + \frac{1}{2} & \frac{1}{2} - \epsilon \le \alpha \le \frac{1}{2} + \epsilon \\ 1 & \frac{1}{2} + \epsilon < \alpha \le 1. \end{cases}$$

Essentially the predictor outputs 0 if α is above $1/2 + \epsilon$, 1 if α is below $1/2 - \epsilon$, and otherwise outputs 0 or 1 probabilistically.

Example: Let $s_m = 00$ and suppose $N^{LZ}(000) = 11$ and $N^{LZ}(001) = 9$. Thus, $\widehat{P}(X_{n+1} = 0|\mathcal{H}_n) = 12/22 = 0.545 = 1 - \widehat{P}(X_{n+1} = 1|\mathcal{H}_n)$. If $\epsilon = .01$ then the predictor would guess 0 for X_{n+1}; if $\epsilon = .25$ then the predictor would guess 0 for X_{n+1} with probability of 13/22 and 1 with probability 9/22.

Compare the output of Vitter and Krishnan [35] with Feder et al's algorithm for this example assuming a single prediction is desired ($r = 1$). The former simply calculates the probability estimates $\widehat{P}(X_{n+1} = 0|\mathcal{H}_n) = 12/22$ and $\widehat{P}(X_{n+1} = 1|\mathcal{H}_n) = 10/22$, so that the prediction is 0. The latter provides predictions with certainty only if the probability estimates for 0 and 1 are not too close (i.e., not within 2ϵ of each other).

If the predictor is used sequentially then Feder, et al showed that its asymptotic error rate is the best possible among predictors with finite memory.

3. Krishnan and Vitter generalized Feder, et al's procedure and result to arbitrary sequences generated from a bigger alphabet [20]; i.e., $|\mathcal{A}| \ge 2$. Their scheme for computing $\widehat{P}(X_{n+1} = a|\mathcal{H}_n)$ for each $a \in \mathcal{A}$ is not only based on how frequently a followed s_m but also on the order in which the symbols followed s_m. Specifically, they assigned probability estimates in the following manner. Suppose after the first occurrence of s_m, the next occurrence is $s_m h_1$ for some symbol h_1, the following occurrence is $s_m h_2$, etc. Consider all the symbols h_i that have followed s_m (after its first occurrence), and create the sequence $h = h_1 h_2 \ldots h_t$. Let $h(i, j)$ denote the subsequence $h_i h_{i+1} \ldots h_j$. Let $q = \lceil \sqrt[4]{t} \rceil$ and $h' = h(4^{q-1} + 2, t)$. Then for each $a \in \mathcal{A}$,

$$\widehat{P}(X_{n+1} = a|\mathcal{H}_n) = \frac{(N(a, h'))^{2^q}}{\sum_{a \in A}(N(a, h'))^{2^q}}. \tag{5}$$

If r predictions are allowed for X_{n+1}, then the predictor uses these probability estimates to choose without replacement r symbols from $\mathcal{A}$.

Example: Suppose 9 symbols from $\mathcal{A} = \{0, 1, 2\}$ followed s_m and the sequence of the symbols is $h = 210011102$. Thus, $q = 2$. The relevant subsequence h' for predicting X_{n+1} is 1102. The frequency of 0, 1 and 2 in the subsequence are 1, 2, and 1 respectively so their probability estimates are $1/18, 16/18$ and $1/18$ respectively. The predictor will pick r of these symbols without replacement using these probabilities.

4. In [7], Bhattacharya and Das proposed a heuristic modification to the construction of the LZ tree, as well as a way of using the modified tree to predict the most likely cells that the user will reside in so as to minimize paging costs to locate the user. The resulting algorithm is called *LeZi-Update*. Although their application (like that of Yu and Leung [37]) lies in the mobile wireless environment, the core prediction algorithm itself is not specific to this domain. For this reason, and for ease of exposition, we include it in this section.

As pointed out earlier, not every substring in $\mathcal{H}_\setminus$ forms a leaf s_i in the LZ parsing tree. In particular, substrings that cross boundaries of the s_i, $0 < i \leq m$, are missed. Further, previous LZ-based predictors take into account only the occurrence statistics for the prefixes of the leaves s_i. To overcome this, the following modification is proposed. When a leaf s_i is created, all the proper suffixes of s_i are considered (i.e., all the suffixes not including s_i itself.) If an interior node representing a suffix does not exist, it is created; and the occurrence frequency for every prefix of every suffix is incremented.

Example. Suppose the current leaf is $s_m = bde$ and the string de is one that crosses boundaries of existing s_i for $1 \leq i < m$. (See Fig. 2). Thus de has not occurred as a prefix or a suffix of any s_i, $0 < i < m$. The set of proper suffixes of s_m is $S_m = \{\gamma, e, de\}$, and since there is no interior node for de, it is created. Then the occurrence frequency is incremented for the root labelled γ, the first-level children b and d, and the new interior node de.

We observe that this heuristic only discovers substrings that lie within a leaf string. Also, at this point it would be possible to use the modified LZ tree and apply one of the existing prediction heuristics e.g., use Eq. 3 and the Vitter-Krishnan method.

However, in [7] a further heuristic is proposed to use the modified LZ tree for determining the most likely locations of the user. This second heuristic is based on the Prediction by Partial Match (PPM) algorithm for text compression [4]. (The PPM algorithm essentially attempts to "blend" the predictions of several order-k Markov predictors, for $k = 1, 2, 3, ...$; we do not describe it here in detail.) Given a leaf string s_m, the set of proper suffixes S_m is found. Observe that each element of S_m is an interior node in the LZ tree. Then, for each suffix, the heuristic considers the subtree rooted at the suffix and finds all the paths in this subtree originating from the root. (Thus these paths would be of length l for $l = 1, 2, \ldots t$, where t is the height of the suffix in the LZ tree.) The PPM algorithm is then applied. PPM first computes the predicted probability of each path in the entire set of paths and then uses these probabilities to compute the most probable symbol(s), which is the predicted location of the user.

5. Yu and Leung [37] use LZ prediction methods for call admission control and bandwidth reservation. Their mobility prediction approach is novel in that it predicts both the location and handoff times of the users. Assume time is discretized into slots of a fixed duration. The movement history $\mathcal{H}_n$ of a user is recorded as a sequence of ordered pairs $(S, l_1), (T_2, l_2), \ldots, (T_n, l_n)$ where S is the time when the call was initiated at cell l_1 and $S + T_i$ is when the ith handoff occurred to cell l_i for $i \geq 2$. In other words, T_i is the relative time (in time slots) that has elapsed since the beginning of the call when the ith handoff was made.

   Like LeZi-Update, if the user is currently at cell l and time $S + T$, the predictor uses the LZ tree to determine the possible paths the user might take and then computes the probabilities of these paths. Unlike LeZi-Update, the computation is easier and is not based on PPM. The algorithm estimates the probabilities $P_{i,j}(T_k)$, the probability that a mobile in cell i will visit cell j at timeslot $S + T_k$, by adding up the probabilities of the paths in the LZ tree that are rooted at the current time-cell pair and contain the ordered pair (j, T_k).

3.3 Other approaches

Chan et al [10] suggest a different approach for location prediction based on using an order-2 Markov predictor with Bayes' Rule. The idea is to first predict the general direction of movement and then use that to predict the next location. For the order-2 predictor the last two terms of the user's itinerary, $L =< L_1, L_2 >$ are used. First the most likely location m steps away from the current location, i.e., L_{2+m}, is predicted based on the user's past history. Then the next location L_3 is predicted using Bayes' Rule and the reference point L_{m+2} by choosing the location B_x with the highest probability as follows.

$$P(L_1L_2B_x|L_{2+m}) = \frac{P(L_{2+m}|L_1L_2B_x)P(L_1L_2B_x)}{\sum_{j=1}^{n} P(L_1L_2B_j)P(L_{2+m}|L_1L_2B_j)} \tag{6}$$

4 Domain-specific heuristics

In this section we discuss several location prediction algorithms that have been proposed for specific application domains.

4.1 Mobile Motion Prediction (MMP)

Liu and Maguire [22] present a location prediction algorithm that can be used for improving mobility management in a cellular network. The movement of a user is modelled as a process $\{M(a,t) : a \in \mathcal{A}, t \in T\}$ where $\mathcal{A}$ is the set of possible locations (called *states*) and T is an index set indicating time. It is assumed that the user's movement is composed of a regular movement process $\{S(a,t)\}$ and a random movement process $\{X(a,t)\}$.

A location is called a *stationary state* if the user resides there longer than some threshold time interval, and a *transitional* state otherwise. A location at the geographical boundary of the service area is called a *boundary* state. For convenience we call the stationary and boundary states *marker* states. Two types of movement patterns are then defined. A Movement Circle (MC) is a sequence of locations that begins and ends with the same location and contains at least one marker state. A Movement Track (MT) is a sequence of locations that begins and ends with a marker state. It is possible for an MC to be an MT and vice versa. It is assumed that the regular movement process $\{S(a,t)\}$ consists only of the MC process $\{MC(a,t)\}$ and the MT process $\{MT(a,t)\}$. The random movement process is further assumed to be a pure (i.e., order-1) Markov process.

The Mobile Motion Prediction (MMP) algorithm consists of a Regularity Detection Algorithm (RDA) that builds up a database of MC and MT seen for each user over time, and a Motion Prediction Algorithm (MPA) that uses this database. Although the details of these algorithms are not clearly specified, it appears that MPA operates as follows. (For convenience we describe the algorithm using MTs, although the process for MCs is similar.) Suppose the most recent $k-1$ locations of the mobile's history are the sequence $L = l_1 l_2 \ldots l_{k-1}$, i.e., L is the suffix of $\mathcal{H}$ of length $k-1$. Suppose there exists an MT stored in the database, $C = c_0 \ldots c_n$, where c_0 and c_n are marker states. Using a matching algorithm (described below), suppose L matches C; we call C a candidate MT. If the current location of the mobile, l_k equals that predicted by C, then C continues as the candidate MT and MPA uses it for prediction (as described below).

Otherwise, MPA uses the matching algorithm on the sequence $L' = l_i l_{i+1} \ldots l_{k-1} l_k$, where l_i is the most recent marker state in L, to find a new MT candidate D.

The matching algorithm uses three matching heuristics. The first is called *state matching* and computes a state matching index μ indicating the degree of similarity in the locations of the mobile's actual itinerary compared to the candidate MT. Using the notation above, for the itinerary L, let m, $0 < m < k$, be the number of locations that appear in both L and C. Then $\mu = m/(k-1)$, and higher values indicate a better match. The second heuristic is *time matching* and computes an index η indicating the degree of similarity in the residence times of the mobile in each location for the mobile's itinerary compared to the candidate MT. Let r_i be the time the mobile spends at each location l_i in L, and similarly s_i be the residence time for location c_i in C. Then

$$\eta = \frac{\sum_{i=1}^{k-2} |r_i - s_i|}{\sum_{i=1}^{k-2} |r_i + s_i|} \tag{7}$$

and lower values indicate a better match. The third heuristic is *frequency matching* and computes an index Φ comparing F' and F, where F' is how often the mobile's itinerary appears in a given time period, and F is how often the candidate MT appears over the time period in the database. (Unfortunately, only approximation equations and an example are given for F and F', so this heuristic is quite unclear). Then $\Phi = |(F'/F) - 1|$ and lower values indicate a better match. The matching algorithm applies the three matching heuristics in sequence, so that the final prediction is dependent on μ, η and Φ.

Note that in MMP any itinerary that cannot be classified based on the stored MT and MC is assumed to be a random movement. The MMP algorithm is not clearly specified and lacks a theoretical foundation but does contain interesting ideas in terms of classifying location types (stationary and boundary states) as well as movement patterns (MC, MT) and different matching heuristics applied in sequence. Since it was one of the first attempts at location prediction for mobility management it is often referenced.

4.2 Segment matching

Chan et al [10] use a simplification of the Liu and Maguire algorithm, which they call the Segment Criterion algorithm. Like Liu and Maguire's stationary states, they define stationary cells based on the residence time of the user in the cell. They then partition the individual user's history into segments, where a *segment* is a sequence of cells that starts with a stationary cell and ends with the same or different stationary cell. Thus a segment is similar to an MT in Liu and Maguire except that it applies only to stationary cells; there is no concept of boundary cells.

The prediction algorithm begins constructing a segment as the user moves, i.e., the user's itinerary after k moves is $L = l_1 l_2 \ldots l_k$ where l_1 is a stationary cell. L is compared with the user's stored segments. A match is found if $l_i = c_i$, $1 \leq i \leq k$, for some stored candidate segment $C_n = c_1 c_2 \ldots c_n$. In that case the prediction is the cell c_{k+1}. If there are multiple candidates then the prediction is the most frequently occurring cell in position $k+1$ among the candidate segments.

Chan et al use two heuristics for overcoming the limitations of relying on the individual user's history. The first heuristic attempts to compensate for sudden changes in movement behavior as follows. The last 10 predictions are compared with the user's actual itinerary; a higher weight is assigned to the latest movement of the user if 6 of the predictions were incorrect, and this weight is decreased gradually if predictions come inside a preset criterion of success. (The weight, the way in which it is decreased, and the criterion are not specified.)

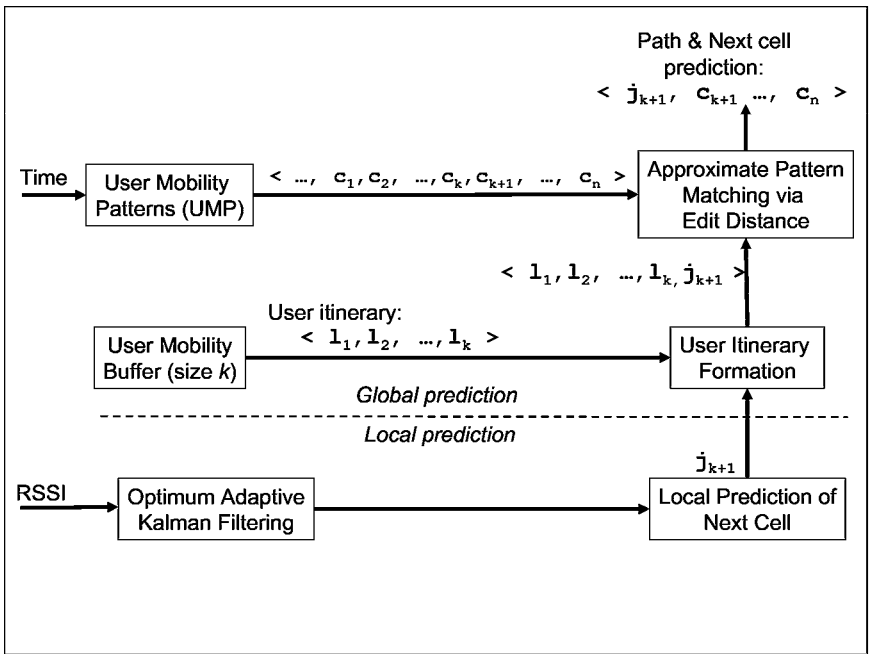


Figure 3: Hierarchical location prediction process [23]

The second heuristic attempts to compensate for users who do not have a movement history, and uses the aggregate history over all users instead. These heuristics are also used for Chan et al's Markov prediction schemes (see sec. 3.1) as well as the probabilistic scheme (sec. 3.3).

4.3 Hierarchical Location Prediction (HLP)

Liu et al [23] have developed a two-level prediction scheme intended for use in mobility management in a wireless ATM environment, but with wider applicability. The lower level uses a Local Mobility Model (LMM) which is a stochastic model for intracell movements, while the top level uses a deterministic model (the Global Mobility Model, or GMM) dealing with intercell movements. The two-level scheme is depicted in Fig. 3 and summarized below.

The local prediction algorithm is intended only for predicting the next cell that the user will visit, while the global prediction can predict the future path. The local prediction algorithm uses consecutive Radio Signal Strength Indication (RSSI) measurements and applies a modified Kalman filtering algorithm to estimate the *dynamic state* of a moving user, where the dynamic state consists of the position, velocity and acceleration of the user. When the user is "close" to a cell boundary, (i.e., in an area called a *correlation area* defined precisely using the geometry of hexagonal cells), the estimated dynamic state is used to determine cell-crossing probability for each neighboring cell, and the cell with the highest crossing probability is output as the predicted next cell. This prediction is used as input to the global prediction algorithm.

Like the Liu and Maguire MMP algorithm, the global prediction algorithm relies on a number of User Mobility Patterns (UMP) recorded for each user. The user's itinerary so far, along with the next cell predicted by the local prediction algorithm, is compared to these stored UMPs and an *edit distance* is generated, which is based on the smallest number of cell insertion, cell deletion, and cell ID modification operations required to make the itinerary

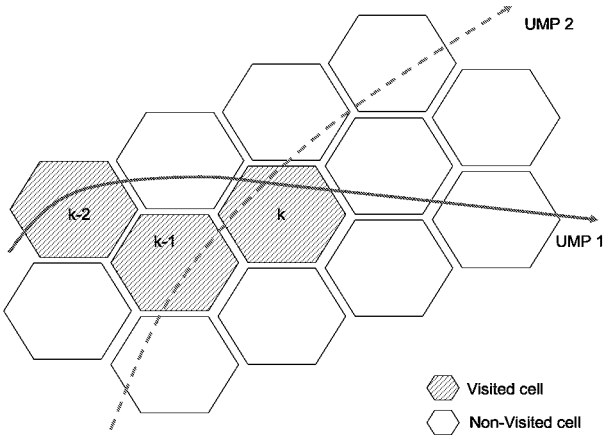


Figure 4: Benefit of local prediction for selecting a candidate UMP [23]

identical to a UMP. If the edit distance is less than a threshold value, the UMP with the smallest edit distance is found using a dynamic programming method; this UMP is assumed to be the candidate UMP and to indicate the general direction of user movement. The remaining portion of the candidate UMP is output as the predicted path for the user.

Liu et al show by simulation that their scheme has a better prediction accuracy than MMP for mobility patterns with a moderate or high degree of randomness.

It is also worth noting that unlike the MMP algorithm the accuracy of next cell prediction using the local prediction algorithm is based purely on RSSI measurements and is independent of the long-term movement patterns of the mobile. This local prediction can be used to improve the path prediction as depicted in Fig. 4. Next cell prediction can help choose between two candidate UMPs when the user's itinerary (shown by the shaded cells) is equidistant in terms of edit distance from them.

4.4 Other approaches

One approach we have not considered so far is to take the help of the user for prediction. Obviously as little burden as possible should be placed on the user, but one could envision a situation where the user is only prompted for her current destination (which could be collected e.g. via a voice prompt) and this is used (possibly along with history information) to do cell and path prediction. Another variation could be that at the start of the day the user is prompted for a list of destinations she will visit during the course of the day, so that interaction is minimized further. Madi et al [25] have developed schemes for prompting for user destination input in this way, although no prediction is carried out as such.

Biesterfeld et al [8] propose using neural networks for location prediction. They have considered both feedback and feed-forward networks with a variety of learning algorithms. Their preliminary results indicated, somewhat non-intuitively, that feed-forward networks delivered better results than feedback networks.

Das and Sen [13] consider how to use location prediction to assign cells to location areas so as to minimize mobility management cost (i.e., the combined paging and location update cost), where the location areas are arranged hierarchically. The assignment is dynamic and is calculated periodically, every τ seconds. The user's movement history is assumed only to consist of a set $\mathcal{L}_c = \{(l_i, f_i) : 0 \leq i \leq c\}$ recording the frequency f_i with which each cell l_i

is visited during the previous period, where c is the number of distinct cells visited.

Then the probability of the user visiting a given cell l_i is calculated simply as the relative frequency with which the user has visited that cell in the previous period, i.e., $f_i / \sum_j f_j$. (Similarly frequencies for visiting areas where two or more cells overlap are recorded and the probability of visiting the overlapping area calculated.) We observe that this scheme is similar to order-0 Markov prediction as described in [7].

These cell visit probabilities are used to assign cells to a *most probable location area* (MPLA). However, it is possible that the user has left this area and moved to an adjoining area, called the future probable location area (FPLA). If the user is not found in the MPLA the FPLA is paged. The cells belonging to the FPLA are determined based on the current mean velocity, the last cell visited, and (optionally) the direction of future movement. Given the cells in the FPLA, the probability that the user will visit a particular cell is estimated by a heuristic that takes into account the total number of cell crossings $\sum_j f_j$, the frequency $\max_i f_i$, and c, the number of distinct cells visited.

5 Conclusions

In this chapter we have provided an overview of different approaches to predicting the location of users in a mobile wireless system. This chapter is not intended to be a comprehensive survey, and in particular we have only summarized a few of the approaches being used in domain-specific algorithms for location prediction.

We see two general ideas pursued in the literature: domain-independent algorithms that take results from Markov analysis or text compression algorithms and apply them to prediction, and domain-specific algorithms that consider the geometry of user motion as well as the semantics of the symbols in the movement history. Domain-specific algorithms tend to have well-founded theoretical principles on which they are based and can make analytical statements about their prediction accuracy. However, in some cases, these statements refer to the asymptotic optimality of their accuracy, i.e., that as the input history approaches infinite length no similar prediction algorithm can do any better. While satisfying from a theoretical point of view, it is unclear how relevant these results are in practice. On the other hand, some domain-specific algorithms offer heuristics that appear intuitively appealing but have no explicit theoretical analysis to support them. Clearly a better bridge between engineering intuition and theoretical analysis would be helpful.

One of the major barriers to practical advancement in this area is the lack of publicly-available empirical data to guide future research. Most studies have used artificial mobility models; relatively few, e.g. [10] have collected empirical data for the domain of interest (cellular handoffs) and used them for validation. We compare the situation to the early work done on caching disk pages in computer systems. A large variety of cache replacement policies, many of which were intuitively plausible, were proposed. It was only empirical data from page fault traces that enabled the conclusion that the Least Recently Used (LRU) algorithm offered the best compromise between simplicity and effectiveness in most cases. Large-scale statistical data for the domains of interest is sorely needed to help provide benchmarks and directions for future research.

Acknowledgements.
We thank Prof. John Kieffer for interesting and helpful discussions, as well as Dr. Xiaoning

He for comments on a draft of this paper.

References

[1] I. Akyildiz, J. Ho, and Y. Lin, *Movement based location update and selective paging for PCS networks*, IEEE/ACM Transactions on Networking **4** (1996), no. 4, 629–638.

[2] I. F. Akyildiz, J. McNair, J. S. M. Ho, H. Uzunalioglu, and W. Wang, *Mobility management for next generation wireless systems*, Proc. IEEE (1999), 1347–1384.

[3] A. Bar-Noy, I. Kessler, and M. Sidi, *Mobile users: To update or not to update?*, ACM/ Baltzer Journal on Wireless Networks **1** (1995), no. 2, 175–195.

[4] T. C. Bell, J. G. Cleary, and I.H. Witten, *Text compression*, Prentice Hall, 1990.

[5] V. Bharghavan and M. Jayanth, *Profile-based next-cell prediction in indoor wireless LAN*, Proc. IEEE SICON (Singapore), Apr. 1997.

[6] V. Bharghavan, K.-W. Lee, S. Lu, S. Ha, J.-R. Li, and D. Dwyer, *The TIMELY adaptive resource management architecture*, IEEE Pers. Comm. (1998), 20–31.

[7] A. Bhattacharya and S.K. Das, *Lezi-update: An information-theoretic framework for personal mobility tracking in PCS networks*, ACM/Kluwer Wireless Networks **8** (2002), no. 2–3, 121–135.

[8] J. Biesterfeld, E. Ennigrou, and K. Jobmann, *Location prediction in mobile networks with neural networks*, Proc. Intl. Workshop on App. of Neural Networks to Telecom., June 1997, pp. 207–214.

[9] J. Chan, R. De Silva, S. Zhou, and A. Senivaratne, *A framework for mobile wireless networks with an adaptive QoS capability*, Proc. Mobile Mult. Comm. (MoMuC), Oct. 1998, pp. 131–137.

[10] J. Chan, S. Zhou, and A. Seneviratne, *A QoS adaptive mobility prediction scheme for wireless networks*, Proceedings of IEEE Globecom (Sydney, Australia), Nov. 1998.

[11] C. Chao and W. Chen, *Connection admission control for mobile multiple-class personal communications networks*, IEEE Journal on Selected Areas in Communications **15** (1997), 1618–1626.

[12] K. C. Chua and S. Y. Choo, *Probabilistic channel reservation scheme for mobile pico/microcellular networks*, IEEE Comm. Lett. **2** (1998), no. 7, 195–196.

[13] S. K. Das and S. K. Sen, *Adaptive location prediction strategies based on a hierarchical network model in a cellular mobile environment*, The Computer Journal **42** (1999), no. 6, 473–486.

[14] F. Erbas, J. Steuer, K. Kyamakya, D. Eggesieker, and K. Jobmann, *A regular path recognition method and prediction of user movements in wireless networks*, IEEE Vehic. tech. Conf. (VTC), Oct. 2001.

[15] M. Feder, N. Merhav, and M. Gutman, *Universal prediction of individual sequences*, IEEE Transactions on Information Theory **38** (1992), 1258–1270.

[16] R. Jain, Y.-B. Lin, C. Lo, and S. Mohan, *A caching strategy to reduce network impacts of PCS*, IEEE J. Sel. Areas Comm. (1994).

[17] R. Jain, Y.-B. Lin, and S. Mohan, *Location strategies for personal communications services*, Mobile Communications Handbook, 2nd ed. (J. Gibson, ed.), CRC Press, 1999.

[18] S. Jiang, D. He, and J. Rao, *A prediction-based link availability estimation algorithm for mobile ad hoc networks*, Proc. IEEE Infocom, 2001.

[19] P. Krishna, N. Vaidya, and D. Pradhan, *Static and adaptive location management in mobile wireless networks*, Computer Comm. **19** (1996), no. 4, 321–334.

[20] P. Krishnan and J. Vitter, *Optimal prediction for prefetching in the worst case*, SIAM Journal on Computing **27** (1998), no. 6, 1617–1636.

[21] D. Levine, I. Akyildiz, and M. Naghshineh, *A resource estimation and call admission algorithm for wireless multimedia networks using the shadow cluster concept*, IEEE/ACM Transactions on Networking **2** (1997), 1–15.

[22] G. Liu and G. Maguire Jr., *A class of mobile motion prediction algorithms for wireless mobile computing and communications*, ACM/Baltzer Mobile Networks and Applications (MONET) **1** (1996), no. 2, 113–121.

[23] T. Liu, P. Bahl, and I. Chlamtac, *Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks*, IEEE J. Sel. Areas Comm. **16** (1998), no. 6, 922–936.

[24] X. Luo, I. Thng, and W. Zhuang, *A dynamic channel pre-reservation scheme for handoffs with GoS guarantee in mobile networks*, Proceedings of IEEE ICC (Vancouver, Canada), 1999.

[25] M. Madi, P. Graham, and K. Barker, *Mobile computing:Predictive connection management with user input*, Tech. report, Dept. Comp. Sci., Univ. Manitoba, Aug. 1996.

[26] M. Naghshineh and M. Schwartz, *Distributed call admission control in mobile/wireless networks*, IEEE Journal on Selected Areas in Communications **14** (1996), 711–717.

[27] C. Oliveira, J. Kim, and T. Suda, *An adaptive bandwidth reservation scheme for high-speed multimedia wireless networks*, IEEE J. Sel. Areas Comm. **16** (1998), no. 6, 858–874.

[28] E. Pitoura and G. Samaras, *Locating objects in mobile computing*, IEEE Trans. of Knowledge and Database Eng. **13** (2001), no. 4, 571–692.

[29] C. Posner and R. Guerin, *Traffic policies in cellular radio that minimize blocking of handoff calls*, Proceedings of the 11th International Teletraffic Congress (Kyoto, Japan), 1985.

[30] R. Ramjee, R. Nagarajan, and D. Towsley, *On optimal call adimission control in cellular networks*, Proceedings of IEEE Infocom (San Francisco, CA), 1996.

[31] M. Riera and J. Aspas, *Variable channel reservation mechanism for wireless networks with mixed types of mobility platforms*, Proc. IEEE Vehic. Tech. Conf. (VTC), May 1998, pp. 1259–1263.

[32] C. Rose, *Minimizing the average cost of paging and registration: A timer-based method*, Wireless Networks **2** (1996), no. 2, 109–116.

[33] N. Shivakumar, J. Jannink, and J. Widom, *Per-user profile replication in mobile environments: Algorithms, analysis, and simulation results*, ACM/Baltzer Mobile Networks and Applications (MONET) **2** (1997), no. 2, 129–140.

[34] W. Su, S. Lee, and M. Gerla, *Mobility prediction and routing in ad hoc wireless networks*, Intl. J. Net. Mgmt. **11** (2001), 3–30.

[35] J. Vitter and P. Krishnan, *Optimal prefetching via data compression*, Journal of the ACM **43** (1996), no. 5, 771–793.

[36] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez, *Cost and imprecision in modeling the position of moving objects*, Proc. IEEE Intl. Conf. Data Eng. (ICDE), Feb. 1998.

[37] F. Yu and V. Leung, *Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks*, Computer Networks **38** (2002), 577–589.

[38] T. Zhang, E. van den Berg, J. Chennikara, P. Agrawal, J-C. Chen, and T. Kodama, *Local predictive reservation for handoff in multimedia wireless IP networks*, IEEE Journal on Selected Areas in Communications **19** (2001), 1931–1941.

[39] J. Ziv and A. Lempel, *Compression of individual sequences via variable-rate coding*, IEEE Transactions on Information Theory **24** (1978), no. 5, 530–536.