

# LeZi-Update: An Information-Theoretic Approach to Track Mobile Users in PCS Networks\*

Amiya Bhattacharya Sajal K. Das

Center for Research in Wireless Computing (CRew)  
Department of Computer Science  
University of North Texas  
Denton, TX 76203-1366  
E-mail : {amiya,das}@cs.unt.edu

## Abstract

The complexity of the mobility tracking problem in a cellular environment has been characterized under an information-theoretic framework. Shannon's entropy measure is identified as a basis for comparing user mobility models. By building and maintaining a dictionary of individual user's path updates (as opposed to the widely used location updates), the proposed adaptive on-line algorithm can learn subscribers' profiles. This technique evolves out of the concepts of lossless compression. The compressibility of the variable-to-fixed length encoding of the acclaimed Lempel-Ziv family of algorithms reduces the update cost, whereas their built-in predictive power can be effectively used to reduce paging cost.

## 1 Introduction

Seamless and ubiquitous connectivity is the strongest driving force today in designing a Personal Communication Service (PCS) network environment. The need to track down a mobile user for satisfying these connectivity requirements leads to what is commonly known as the *mobility tracking* or *location management* problem. In short, while enjoying the freedom of being mobile, the user creates an *uncertainty* about the exact location of the registered portable device or *mobile terminal* (henceforth called just *mobile*) being carried. The collector network deployed by the service provider has to work against this uncertainty for a successful call delivery. Furthermore, this has to be done effectively and efficiently for every PCS subscriber. To avoid dropping, a call must be routed to the recipient within an allowable time constraint, yet with as less information exchange as possible. Transfer of any more message than necessary results in wastage of valuable resources such as scarce wireless bandwidth and mobile equipment power, thus increasing the operational cost that the user has to bear ultimately.

\*This work is supported by grants from Texas Advanced Research Program (under award no. TARP-003594-013) and Nortel Networks at Richardson, Texas.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Mobicom '99 Seattle Washington USA

Copyright ACM 1999 1-58113-142-9/99/08...\$5.00

The wireless network for a PCS system is still built upon an underlying *cellular* architecture. The service area is divided into a collection of *cells*, each serviced by a *base station* (BS). Several BSs are usually wired to a *base station controller* (BSC), and a number of BSCs are further connected to a *mobile switching center* (MSC) forming a tree-like cluster. This hierarchical wired connection of the MSC, BSCs and BSs, along with the air-link between the BSs and the mobiles form the *collector network*. The *backbone* of the PCS network consists of the existing wire-line networks (such as PSTN, ISDN and the Internet) interconnecting the collector networks. The MSCs act as the gateway for the collector network to the backbone.

As a whole, location management involves two kinds of activities – one on the part of the system and the other on the part of the mobile. On a call arrival, the system initiates a search for the target mobile, by simultaneously polling all the cells where it can possibly be found. The MSC broadcasts a *page* message over a designated *forward control channel* via the BSs in a set of cells where the mobile is likely to be present. All the mobiles listen to the page message and only the target sends a response message back over a *reverse channel*. This search mechanism is called *paging*. In case no information about the mobile is available, the system may have to page all the cells in the service area. As the PCS providers are shooting for larger and larger (even continent-wide) service area, the paging-only location tracking turns out to be inadequate. If an exhaustive search is performed for each and every call that arrives, the signaling traffic would become enormous even for moderately large network [20]. The limited number of paging channels are bound to overload as the call volume grows. To put an upper bound on the amount of *location uncertainty*, a mobile is made to report from time to time. This reporting, called a *location update* or *registration*, effectively limits the search space for paging at a later point of time. The registration mechanism starts with an *update message* sent by the mobile over a *reverse channel*, which is followed by some traffic that takes care of related database operations.

In this paper, we look at the location management problem and its complexity from an information-theoretic viewpoint. This outlook provides the insight to design an adaptive on-line algorithm for tracking a mobile, which is optimal in terms of both update and paging. The object of our update scheme is to learn user mobility with optimal message exchange. Learning endows the paging mechanism with a predictive power which reduces average paging cost. In Section 2, we survey the existing location management

techniques. Section 3 explains the motivations behind our work and the principles it is based on. In section 4 we look for general and flexible models for both the cellular network and an individual user mobility. Section 5 shows how to compare mobility models in terms of uncertainty, and discusses the richness and limitation of the models. Finally, in Section 6, we present the new adaptive and on-line protocol “LeZi-update”, designed around the well-known universal lossless compression techniques.

## 2 Location Management: A Taxonomy of Approaches

### 2.1 Paging

The naive approach to location tracking is to page the mobile simultaneously either in the entire network, or in a part as dictated by the last update message. As pointed out by Rose and Yates [24], this may be an overkill. In reality, paging must be completed within an allowable *delay constraint*, and there exists the possibility of sequentializing the procedure. An alternative considered there is to split the service area into *location areas* (LAs), where all the cells within an LA are paged simultaneously. When a call arrives, the LAs are paged sequentially for the mobile following an ordering termed as *paging strategy*. A very intuitive result derived in [24] states that, under steady-state location probability distribution of a mobile user, the optimal paging strategy (in terms of the mean cost of paging) with no delay constraint should page the LAs in the order of decreasing probability values. Clearly, a uniform distribution is the worst adversary, because no additional improvement is obtainable by changing paging strategy. An algorithm to find the optimal paging sequence under given delay constraints is also presented in [24]. However, the worst case still ends up with a system-wide paging and incurring high cost.

To counteract the worst case complexity issue in pure paging based strategies, most location management schemes rely on time to time *location updates* by the mobile. The idea is to enforce an upper bound on the uncertainty by restricting the paging domain within the vicinity of the last known location. The cost of location update, however, adds a new component to the location management cost. The update cost over a time period is proportional to the number of times the mobile updates, whereas the paging cost depends on the number of calls arrived and the number of cells paged while routing each of those calls. The trade-off between the two components is evident from the fact that, the more frequently the mobile updates (incurring higher update cost) the less is the number of paging attempts required to track it down.

From an operational point of view, it seems that paging is more fundamental than update in a cellular system. Yet, it is somewhat interesting to observe that the majority of the research on location management has actually focussed on update schemes, assuming some obvious version of paging algorithm. Seemingly, either the above-mentioned result in [24] has not received enough attention, or it has not become apparent how to obtain a probability model for users passage that is good enough for use in practice. As pointed out in [32], not enough studies have been conducted in realistic movement profiles of mobiles that take care of speed and directional variations. Usually, pedestrian and vehicular movement are treated separately. This makes the mobility models inappropriate for PCS networks [7] and third-generation systems [16].

Most paging algorithms proceed with a high reliance fac-

tor on the latest update information. The mobile’s last known position and its surroundings are considered to be the most probable current position – the probability decreasing in an omni-directional way with increasing distance. This is the underlying assumption for the popular *cluster paging* [19] and *selective paging* [1, 2, 13]. Bar-Noy, Kessler and Sidi [4], as well as Birk and Nachman [6] have taken into account the directional bias in user movement by associating a number of states for each cell under a Markov model. Only Pollini and I [21] have considered user profile in designing paging algorithm. They assumed that probabilistic information about profile is readily available either with user or at billing database.

### 2.2 Static Update Schemes

The following two location update schemes, based on either partitioning of cells into location areas (LA) or selection of a few designated reporting cells, have been characterized as *static* or *global* techniques [4]. These are static schemes because the cells at which a mobile updates are fixed. They are also global in the sense that all or a batch of mobiles always originate their update messages from the same set of cells.

#### 2.2.1 LA partitioning

The update scheme most widely adopted by current cellular systems (such as GSM [18]), follow the idea presented in [24]. The service area under an MSC is partitioned into location areas (LA), that are formed by non-overlapped grouping of neighboring cells. A mobile must update whenever it crosses an LA boundary. Its location uncertainty is reduced by limiting the search space to the set of cells under the current LA. All the cells under the LA are paged upon a call arrival, resulting in an assured success within a single step. The base stations must broadcast the LA-id (along with the cell-id) to aid the mobiles follow the update protocol. Consequently, a global LA assignment is induced for all subscribers. An obvious drawback of this scheme is that the update traffic originate only in the boundary cells of the LAs, thereby reducing the bandwidth availability for other calls.

Xie, Tabbane and Goodman [30] have shown how to partition cells into optimal location areas. Assuming that the relative cost of paging vs. update and how the paging cost varies with LA size, are known, their solution is dictated by the call arrival rate and mobility of the user. Two variants of the method are proposed. The pure *static* variant uses the average call arrival rate and mobility index for all users. Thus, a mobile behaving as a potential outlier would transmit frequent un-informative update messages by crossing LA boundaries. In contrast, a somewhat *dynamic* variant of this primarily static scheme allows different LA assignments for different mobiles, based on individual call arrival and mobility patterns. This alleviates the localization problem of the update traffic to some extent, although at the cost of the computational complexity involved in choosing, maintaining and uploading a wide range of LA maps to individual mobiles. Kim and Lee [14] have taken the dynamism further by optimizing on the signaling costs that reflects the mobile’s direction of movement and regional cell characteristics.

#### 2.2.2 Reporting center selection

An alternative approach due to Bar-Noy and Kessler [3] does not impose any partition on the cellular map, but designate some cells as *reporting cells* where the mobiles must update

upon entering. On arrival of a call, the mobile is paged in the vicinity of the reporting cell it has last updated at. Choosing an optimal set of reporting cells for a general cellular network has been shown to be NP-complete. Optimal or near optimal solutions for special types of cellular topologies (e.g. tree, ring and grid), and an approximation technique for solutions to general graphs have been presented. But the scheme is built upon a number of simplifying assumptions such as square shape for cells and LAs, and a fluid flow model of user mobility.

The basic drawbacks of a static and global scheme, even if reduced by this approach, still lingers. A user can generate un-informative update messages by hopping in and out of reporting cells. As suggested by Das and Sen [11], considering per-user mobility is the first step towards dealing with these problems. Following the spirit of reporting center selection, they impose a selective update scheme tuned to individual users over an LA-based cellular network. Such a technique lies in between the static update schemes and the dynamic ones, described next.

### 2.3 Dynamic update schemes

Under these schemes, the mobile updates based on only the user's activity, but not necessarily at predetermined cells. These are *local* in the sense that mobiles can make the decision whether to update or not without gathering any global or design specific information about the cell planning. Three major schemes fall under the dynamic category, viz. (i) distance-based, (ii) movement-based and (iii) time-based, which are named by the kind of threshold used to initiate an update.

#### 2.3.1 Distance-based

Under the *distance-based* scheme [1, 4, 13, 17], the mobile is required to track the Euclidean distance from the location of the previous update and initiates a new update if the distance crosses a specified threshold  $D$ . Although the distance would ideally be specified in terms of a unit such as mile or kilometer, it can be specified in terms of the number of cells between the two positions. So the location uncertainty is reduced by limiting the search space to the set of cells falling within a circular region of radius  $D$  around the last known cell. One can think of these cells effectively forming a user and time specific LA. However, it is difficult to measure the Euclidean distance between two locations, even if the traversed distance is made available from the vehicle. Irrespective of its characterization among local ones, implementing distance-based scheme calls for some global knowledge about the cell map. In order for the mobile to identify the cells within the distance threshold, the system has to load these cell-ids as a response to its update message. Madhow, Honig and Steiglitz [17] have described how to find an optimal threshold  $\hat{D}$  by minimizing the expectation of the sum of update and paging costs until the next call. Considering the evolution of the system in between call arrivals under a memoryless movement model, an iterative algorithm based on dynamic programming is used to compute the optimal threshold distance. A similar approach is used by Ho and Akyildiz [13] to compute the optimal threshold distance,  $\hat{D}$ , under a two-dimensional random walk user mobility model over hexagonal cell geometry. The random walk is mapped onto a discrete-time Markov chain, where the state is defined as the distance between the mobile's current location and its last known cell. The residing area of

the user contains  $\hat{D}$  layers of cells around its last known cell, which is partitioned into paging areas according to the given delay constraint.

#### 2.3.2 Movement-based

The *movement-based* [2, 4] scheme is essentially a way of over-estimating the Euclidean distance by traversed distance, when the distance is considered only in terms of the number of cells. The mobile needs to count the number of cell boundary crossings and update when the count reaches certain threshold  $M$ . It is truly a local scheme since there is no longer any need for any knowledge about cell neighborhood. The penalty paid is an increased number of updates that it triggers counting local movements between different cells, even though the distant threshold is not being crossed.

#### 2.4 Time-based

Under the *time-based* [4, 25] scheme, the mobile sends periodic updates to the system. The period or time threshold  $T$  can easily be programmed into the mobile using a hardware or software timer. While this makes it a truly local and attractive solution for implementation, the cost due to redundant updates made by stationary mobiles has to be accommodated. A mobile is paged by searching all possible cells reachable by the user within the elapsed time from the last known cell. Thus the search space evolves as a function of the elapsed time, user mobility, and possibly the last known cell.

Bar-Noy, Kessler and Sidi [4] have compared the time-based, movement-based and distance-based update schemes in terms of the paging cost with varying update rates. Two types of user movement models, viz. independent and Markovian, on a ring cellular topology. It has been observed that the distance-based scheme performs the best consistently. The result is intuitive because the distance threshold directly puts an upper bound on the location uncertainty, whereas in both time and movement-based schemes the uncertainty imposed by user mobility interacts with it.

## 3 Mobility Tracking: A New Perspective

This section starts with the motivations behind our work in this paper, revisits three fundamental questions posed in [24, 25], followed by a summary of our contributions and novelty of the underlying approach which help unfold the intrinsic nature and complexity of the location management problem in a mobile computing environment.

### 3.1 Motivations

Our approach to the location tracking problem is primarily motivated by the following observations:

1. The location management problem has been formulated by the vast majority of researchers as some static optimization problem to minimize either paging or update cost, or a combination thereof. The inputs to the optimization problem are often some probability values that need to be computed from gathered statistics from system snapshots. The static version of optimization algorithm works off-line to provide a solution that remains in effect until the next run. This interval is very critical to the performance of the technique, but is not provided by the static optimization formulation.

Educated guesses are used in most cases. Formulating the problem as a dynamic optimization provides a better insight into its stationary behavior and applicability of static versions. Better yet, it may lead to the design of a *competitive on-line algorithm* to solve the problem.

2. Unlike the resource management problems in cellular networks, the location tracking problem is user oriented by definition. Naturally, it would be wise to make use of personal mobility and calling profile of individual subscribers for optimization purposes. Both the update and paging techniques should be user specific to add the “personal” touch to the *personal communication service*. Knowledge representation and learning of user profile are thus two key factors.
3. The effectiveness of sequentializing the paging process critically depends on early success of the deployed paging strategy. Large number of failed paging attempts would not only result in more call drops, but also cause overloading on paging channels. The essence of designing a good paging strategy is to enhance the predictability of a mobile’s behavior making use of the user profile.
4. Since the sole purpose of the update mechanism is to aid the paging process, there is no reason to treat them as two independent components of location management cost. As opposed to deciding on a paging strategy first and then optimizing on the update strategy, one can come up with a collaborative pair of paging and update policies. In other words, the update mechanism needs to keep the system better informed about user’s mobility, sending the maximum possible information in a compact form and avoiding redundancy as far as possible.

### 3.2 Open Questions

Before proceeding further, we first attempt to address and re-interpret the three basic questions posed in [24, 25]:

1. **Complexity:** Given a user mobility model, what is least amount of effort necessary on the average, to track the mobile? What metric do we use to quantify the effort and what unit do we use?
2. **Optimality:** Given that the user mobility model is known to both the user and the system, how to pick an optimal paging strategy, an optimal update scheme, or an optimal combination of paging and update policies?
3. **Learning:** How can the time-varying dynamic location probabilities be efficiently estimated based on measurements taken from user motion? How to pick the right mobility model that does not impose fundamental restriction in terms of richness?

### 3.3 Our Contributions

To the best of our knowledge, we make the first model-independent attempt to identify and characterize the problem complexity of the location management, which is absolutely essential if we are looking for an optimal solution. We relate the complexity of the mobility tracking to the location *uncertainty* of the mobile, which is measurable both in the mathematical and physical sense. From an information theoretic perspective, Shannon’s *entropy* measure is an

ideal choice for quantifying this uncertainty [9, 26]. Entropy captures the uncertainty of a probabilistic source, which is also the information content of the message (e.g. in number of bits) it generates. The performance of the tracking algorithm can be measured too in the number of bits exchanged between the mobile and the system database during update or paging. The algorithm certainly cannot track the mobile by exchanging any less information on the average than the uncertainty created by its mobility, and hence is optimal when these two amounts are the same.

An analogy exists in the field of data compression [5], which says that a message cannot be compressed beyond the entropy of its source without losing any part of it. Optimality is achieved if the length of the compressed message approaches the entropy, i.e. the redundancy in the encoding approaches zero. Motivated by this duality, we have designed an update scheme around a compressor-decompressor duo. A family of efficient on-line algorithms for text compression is known to show asymptotically optimal performance [31]. Some versions of these algorithms are hardware-implementable and are in commercial use (e.g. V.42bis compression standard for modems) [29], which indicates practical viability. It has already been understood that a good data compressor must also be a good predictor, because one must predict future data well to compress them. Such techniques based on compression have earlier been used as predictors in caching and prefetching of database and hypertext documents [10, 27]. Here we have molded them into efficient update and paging techniques for a universal user mobility model in a cellular network environment.

## 4 User Mobility

The user mobility model plays too important a role in designing location management schemes. The underlying model can potentially influence the analysis and simulation results so much, that observations and conclusions would merely reflect the nature of the model itself. Thus, there are reasons to be skeptic and cautious about extrapolating theoretical claims to practice. For example, Markov models are often constructed with states dictated by the current cell in a regular geometric cellular architecture. Transition probabilities for a typical user to move into a specific neighboring cell are arbitrarily assumed. This is far from being practical.

### 4.1 Network topology

Structured graph models for a cellular network have been quite popular among researchers engaged in solving the location management problem. Circular, hexagonal or square areas are often used to model the cells, while various regular graph topologies such as rings, trees, one and two-dimensional grids are used to model their interconnection. These models do not accurately represent a real cellular network, where the cell shapes may vary depending on the antenna radiation pattern of the base stations, and a cell can have an arbitrary (but bounded) number of neighbors. Although structured graphs help in the early stage of frequency planning, they over-simplify the mobility tracking problem. Since enforcing strict assumption on the topology has a potential to weaken the model, we refrain from making any assumption about either the geometry or the interconnections between cells.

With a GSM-like deployment in mind, let us consider an LA-based system. The network can be represented by a bounded-degree, connected graph  $G = (\mathcal{V}, \mathcal{E})$ , where the

node-set  $\vartheta$  represents the LAs and the edge-set  $\varepsilon$  represents the neighborhood (roads, highways etc.) between pairs of LAs. Figure 1 shows a service area with eight LAs, viz.  $a, b, c, d, e, f, g, h$ , and the corresponding graph representation.

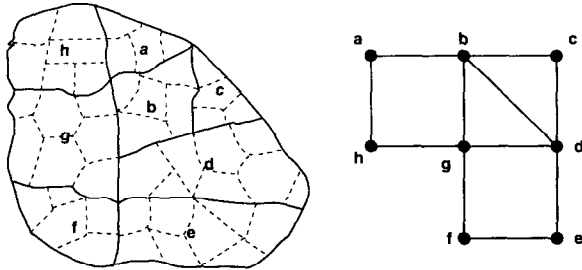


Figure 1: Modeling an actual cellular system

The general graph representation is of course not restricted to only LA-based cell planning. Some existing systems use paging strategies that work with a finer granularity, and choose the individual cells to be paged next. In such cases, our model would give rise to a graph with a larger node-set  $\vartheta = \{a_1, a_2, \dots, b_1, b_2, \dots, c_1, c_2, \dots\}$ , where  $a_1, a_2, \dots$  are cells in LA  $a$  and so on. We introduce a more general term *zone* to refer to a node in our network model. A zone can be an LA or a cell depending on the system. The important thing is that the current zone-id should always be made available to a mobile within by frequent and periodic broadcast messages from the BSs.

## 4.2 Movement history

The real power of an adaptive algorithm comes from its ability to learn. As and when the events unfold, a learning system observes the history to use it as a source of information. Let us now look at the movement history of a typical user within the service area shown in Figure 1. For simplicity, we only record the movement at the coarse granularity of LAs. This means that the user must be in one of the eight zones, viz.  $a, b, c, d, e, f, g, h$ , at any point of time. Suppose the service was turned on at 9:00 a.m., when the mobile initially registered. Table 1 shows the time (limited to the precision of a minute) at which the LA boundaries were crossed. Current time is 9:00 p.m. An update message reports the current zone to the system. Consequently, all that the system obtains from an update scheme is a sequence of zone-id's. The information carried by this sequence about the user's mobility profile depends heavily on the underlying update scheme and how it is interpreted by the system.

Let us compare and contrast the sequences generated by the time-based and movement-based schemes. Table 2 shows the zone sequences generated. For the time-based scheme, two values of the threshold  $T$  have been considered, viz. 1 hr and  $\frac{1}{2}$  hr. Both do a good job in capturing the fact that the user resides in zones  $a$  and  $b$  for a relatively longer time as compared to zones  $c$  and  $d$ . On the other hand, smaller values of  $T$  must be chosen to trace in detail the routes taken by the user. The choice of  $\frac{1}{2}$  hr performed better than that of 1 hr in that respect, yet missed to detect the zone  $c$  in the round trip  $a \rightarrow b \rightarrow c \rightarrow d \rightarrow c \rightarrow b \rightarrow a$ . In contrast, the movement-based scheme is more apt in capturing routes. Two choices of the threshold  $M$ , viz. 1

and 2 have been used to illustrate its effectiveness. Clearly,  $M = 1$  captures movements in the finest detail. Yet, both lack in their capability to convey the relative durations of residence.

Based on these observations we are now motivated to use a combination of time-based and movement-based schemes to make the movement history more informative. The last row of Table 2 shows the zone sequence obtained if a dual mode update is deployed. This scheme generates an hourly update starting at 9:00 a.m., as well as whenever a zone boundary crossing is detected. In the rest of the paper we assume such an update scheme and use this sequence  $aaababbbbaabccddcbaaaa \dots$  for illustrative purposes. However, without any loss of generality, we view the *movement history* as just a sequence of zones reported by successive updates.

**Definition 1** *The movement history of a user is a string " $v_1 v_2 v_3 \dots$ " of symbols from the alphabet  $\vartheta$ , where  $\vartheta$  is the set of zones under the service area and  $v_i$  denotes the zone-id reported by the  $i$ th update. Consequently,  $v_i$ s are not necessarily distinct.*

## 4.3 Mobility model

A clear understanding of the distinctions between the *movement history* and the *mobility model* is necessary before an appropriate definition of the later can be given. The movement history of a user is deterministic, but a matter of the past. The mobility model, on the other hand, is probabilistic and extends to the future. The tacit assumption is that a user's movement is merely a reflection of the patterns of his/her life, and those can be learned over time in either off-line or on-line mode. Specifically, the patterns in the movement history correspond to the user's favorite routes and habitual duration of stay. The essence of learning lies in extracting those patterns from history and storing in the form of knowledge. Learning aids in decision making when re-appearance of those patterns are detected. In other words, learning works because "history repeats itself."

The characterization of mobility model as a probabilistic sequence suggests that it be defined as a stochastic process, while the repetitive nature of identifiable patterns adds stationarity as an essential property. Prior works referenced in this paper also assume this property either explicitly or implicitly.

**Definition 2** *The mobility model of a user is a stationary stochastic process  $\mathcal{V} = \{V_i\}$ , such that  $V_i$  assumes the value  $v_i \in \vartheta$  in the event of the  $i$ th update reporting the user in zone  $v_i$ . The joint distribution of any subsequence  $V_i$ s is invariant with respect to shifts in the time axis, i.e.,*

$$\begin{aligned} \Pr[V_1 = v_1, V_2 = v_2, \dots, V_n = v_n] \\ = \Pr[V_{1+l} = v_1, V_{2+l} = v_2, \dots, V_{n+l} = v_n] \end{aligned} \quad (1)$$

for every shift  $l$  and for all  $v_i \in \vartheta$ . The *movement history* is a trajectory or sample path of  $\mathcal{V}$ .

An important question is why such a general mobility model is not as popular as the restrictive models so abundant in the literature. The most likely reason is that the general model allows nothing to be assumed to start the analysis. The model has to be learned. Given that the user mobility model defined just as a stationary process, learning is possible if and only if one can construct a universal predictor or estimator for this process. Here we illustrate an

Table 1: User movement between 9:00 a.m. and 9:00 p.m.

Time	a.m.			p.m.							
	11:04	11:32	11:57	3:18	4:12	4:52	5:13	6:11	6:33	6:54	...
Crossing	$a \rightarrow b$	$b \rightarrow a$	$a \rightarrow b$	$b \rightarrow a$	$a \rightarrow b$	$b \rightarrow c$	$c \rightarrow d$	$d \rightarrow c$	$c \rightarrow b$	$b \rightarrow a$	...

Table 2: Zone sequence reported by various update schemes

Time-based ( $T = 1$ hr)	aaabbbbacdaaa...
Time-based ( $T = \frac{1}{2}$ hr)	aaaaabbbbbbaabccddcaaaaa...
Movement-based ( $M = 1$ )	abababcdbca...
Movement-based ( $M = 2$ )	aaacca...
Time- & movement-based ( $T = 1$ hr, $M = 1$ )	aaababbbbbaabccddcbaaaa...

approach for building models from history. Rigorous treatments on these topics appear in [12, 22, 23, 28].

Let us first get a better feel of how commonly used models interpret the movement history and end up imposing restrictions during the learning phase.

**Ignorant model:** The ignorant model disbelieves and disregards the information available from movement history. Due to the lack of knowledge, it assigns equal residence probabilities to all the eight zones (see Figure 1), i.e.  $\pi_a = \pi_b = \pi_c = \pi_d = \pi_e = \pi_f = \pi_g = \pi_h = \frac{1}{8} = 0.125$ . The assumption of uniform probability distribution suffers from the consequence that no single paging strategy can be adjudged better than another in terms of average paging cost [24].

**IID model:** The IID model assumes that  $V_i$ s are independent and identically distributed. Using the relative frequencies of the symbols as estimates of residence probabilities, we obtain  $\pi_a = \frac{10}{23} \approx 0.435$ ,  $\pi_b = \frac{8}{23} \approx 0.348$ ,  $\pi_c = \frac{3}{23} \approx 0.13$ ,  $\pi_d = \frac{2}{23} \approx 0.087$ , and  $\pi_e = \pi_f = \pi_g = \pi_h = 0$ .

**Markov model:** The simplest possible Markov model assumes that the process  $S$  is a time-invariant Markov chain, defined by

$$\Pr[V_k = v_k | V_1 = v_1, \dots, V_{k-1} = v_{k-1}] = \Pr[V_k = v_k | V_{k-1} = v_{k-1}] \quad (2)$$

$$= \Pr[V_i = v_i | V_{i-1} = v_{i-1}] \quad (3)$$

for any arbitrary choice of  $k$  and  $i$ .

Zones  $e, f, g$  and  $h$ , never being visited, acquire zero probability mass. The effective state space is thus reduced to the set  $\{a, b, c, d\}$ . The one-step transition probabilities

$$P_{i,j} = \Pr[V_k = v_j | V_{k-1} = v_i],$$

where  $v_i, v_j \in \{a, b, c, d\}$

are estimated by the relative counts. From Figure 2 the probability transition matrix  $\mathbf{P} = ((P_{i,j}))$  is given by

$$\mathbf{P} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 & 0 \\ \frac{3}{8} & \frac{1}{2} & \frac{1}{8} & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

The Markov chain is finite, aperiodic and irreducible, and thus ergodic. Let  $\mathbf{\Pi} = [\pi_a \ \pi_b \ \pi_c \ \pi_d]^T$  be the steady-state

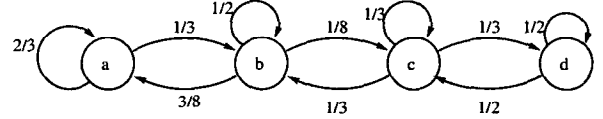


Figure 2: Markov model (order-1) for movement profile

probability vector. Solving for  $\mathbf{\Pi} = \mathbf{\Pi} \times \mathbf{P}$  with  $\pi_a + \pi_b + \pi_c + \pi_d = 1$ , we obtain  $\pi_a = \frac{9}{22} \approx 0.409$ ,  $\pi_b = \frac{4}{11} \approx 0.364$ ,  $\pi_c = \frac{3}{22} \approx 0.136$ ,  $\pi_d = \frac{1}{11} \approx 0.091$ . Also,  $\pi_e = \pi_f = \pi_g = \pi_h = 0$ .

**Finite-context model:** The ignorant model is incapable of learning and cannot lead to any adaptive technique. The IID model takes the first step towards learning from movement history. For example,  $\langle a, b, c, d, e, f, g, h \rangle$  is an optimal unconditional paging strategy that can be derived based on the IID model. However, if we know that the mobile has made the last update in zone  $d$ , neither  $a$  nor  $b$  is a more likely candidate for paging in comparison to  $c$  or  $d$ . Unfortunately, the IID model does not carry any information about the symbols' order of appearance and falls short in such situations. The Markov model carries a little more information about the ordering, at least to the extent of one symbol context. To be more precise, let us adopt the terminology *order-1 Markov model* to refer to this particular model. The same nomenclature designates the IID model as the *order-0 Markov model*. To maintain the sequence, the concept of order has to be extrapolated even to the negative domain without much real meaning. We call the ignorant model an *order-(-1) Markov model*.

The construction of higher order Markov models is illustrated by enumerating all order-2 contexts in the sequence "aaababbbbbaabccddcbbaa" and the symbols that appear in those contexts with their respective frequencies. Table 3 enumerates the contexts with symbol frequencies for orders 0, 1 and 2. An entry of " $v|w(f)$ " implies that the symbol  $v \in \mathcal{V}$  appears with frequency  $f$  in the context  $w \in \mathcal{V}^*$ <sup>1</sup>. Null contexts have not been shown explicitly. In other words,  $f$  is the number of matches of " $w.v$ " in the history, where the *dot* represents concatenation. A dictionary of such contexts can be maintained in a compact form by a *trie* or *digital search tree*, as shown in Figure 3. Every node represents a context, and stores its last symbol along with the rela-

<sup>1</sup>Regular grammar notation for a sequence of zero or more symbols from set  $\mathcal{V}$

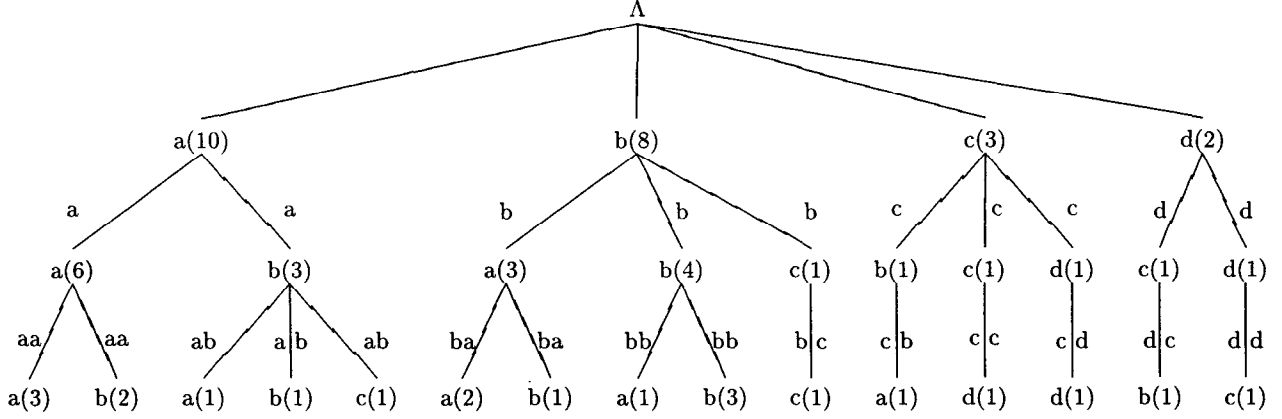


Figure 3: A trie for all contexts up to order-2 in “aaababbbbaabccddcbaaaa...”

Table 3: Contexts of orders 0, 1 and 2 with frequencies

Order-0	Order-1		Order-2		
a(10)	a a(6)	b c(1)	a aa(3)	a ba(2)	a cb(1)
b(8)	b a(3)	c c(1)	b aa(2)	b ba(1)	d cc(1)
c(3)	a b(3)	d c(1)	a ab(1)	a bb(1)	d cd(1)
d(2)	b b(4)	c d(1)	b ab(1)	b bb(3)	b dc(1)
	c b(1)	d d(1)	c ab(1)	c bc(1)	c dd(1)

tive frequency of its appearance at the context of the parent node. Naturally, a node can have at most  $|\mathcal{V}|$  children. The root, at level 0, represents a null context. Level  $l$  stores the necessary statistics for order- $(l - 1)$  Markov model.

It is now intuitively clear how higher order Markov models carry more detailed information about the ordering of symbols in a sequence. Applied to the update sequences, they capture a lot of information about the user’s favorite routes. For example, according to the order-1 model, the probability of the route “abcbcd” being taken is  $\frac{9}{22} \times \frac{1}{3} \times \frac{1}{8} \times \frac{1}{3} \times \frac{1}{8} \times \frac{1}{3} = \frac{1}{4224} \approx 2.37 \times 10^{-4}$ . However small this may seem, such a zigzag route is highly unlikely to be taken by any sensible person, and should have been assigned a zero probability. Under the order-2 model, this route turns out to be an impossible event, as expected.

The possibility of Markov models based on higher order finite contexts adds hope as well as confusion to the pursuit of a general mobility model. Does an increase in the order of context always make the model richer? There should be a limit to the richness, because the sought after general model has to be the richest. At what order  $k$ , do we stop? Should we use order- $k$  model only, or consider all models of orders 0 through  $k$  inclusive? And, what is the figure of merit for richness anyway?

## 5 Location Uncertainty and Entropy

It seems impossible to answer the questions posed in the previous section without a quantitative comparison of the candidate models. Of course, we need a fair basis for such comparison, and uncertainty is a potential choice. The rule of thumb is that the lower the uncertainty under a model, the richer the model is. Consequently, we need to formalize the notion of uncertainty we have been talking about since

the beginning.

### 5.1 Basic terminologies

We have defined user mobility earlier as a stochastic process  $\mathcal{V} = \{V_i\}$ , where  $V_i$ s form a sequence of random variables. The traditional information-theoretic definitions of *entropy* and *conditional entropy* of random variables, as well as *entropy rate* of a stochastic process are given below [9, 26].

**Definition 3** The *entropy*  $H_b(X)$  of a discrete random variable  $X$ , with probability mass function  $p(x)$ ,  $x \in \mathcal{X}$ , is defined by  $H_b(X) = -\sum_{x \in \mathcal{X}} p(x) \log_b p(x)$ . The limiting value “ $\lim_{p \rightarrow 0} p \log_b p = 0$ ” is used in the expression when  $p(x) = 0$ . The base of the logarithm depends on the unit used. As we usually measure information in terms of bits, using  $b = 2$  we write

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \lg p(x). \quad (4)$$

Since probability  $p(x) \in [0, 1]$ , we see that  $H(X) \geq 0$ .

**Definition 4** The *joint entropy*  $H(X, Y)$  of a pair of discrete random variables  $X$  and  $Y$  with a joint distribution  $p(x, y)$ ,  $x \in \mathcal{X}$ ,  $y \in \mathcal{Y}$ , is defined by

$$H(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \lg p(x, y). \quad (5)$$

Also the *conditional entropy*  $H(Y|X)$  is defined as

$$\begin{aligned} H(Y|X) &= \sum_{x \in \mathcal{X}} p(x) H(Y|X = x) \\ &= -\sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x) \lg p(y|x) \end{aligned} \quad (6)$$

$$= -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \lg p(y|x) \quad (7)$$

**Definition 5** The *per symbol entropy rate*  $H(\mathcal{V})$  for a stochastic process  $\mathcal{V} = \{V_i\}$ , is defined by

$$H(\mathcal{V}) = \lim_{n \rightarrow \infty} \frac{1}{n} H(V_1, V_2, \dots, V_n) \quad (8)$$

if the limit exists. The conditional entropy rate  $H'(\mathcal{V})$  for the same process is defined by

$$H'(\mathcal{V}) = \lim_{n \rightarrow \infty} H(V_n | V_1, V_2, \dots, V_{n-1}) \quad (9)$$

if the limit exists.

## 5.2 Comparison of models

First we try to compare different models based on the two entropy rates  $H(S)$  and  $H'(S)$ . Let us consider the three special cases of context models of orders -1, 0 and 1. Let us use the notation  $\Pr[V_1 = v_1, V_2 = v_2, \dots, V_n = v_n] = p(v_1, v_2, \dots, v_n)$  for all  $v_i \in \vartheta$ .

**Order-(-1) model:**  $V_i$ s are independent and uniformly distributed, i.e. with distribution  $p(v) = \pi_v = \frac{1}{8}$ ,  $\forall v \in \vartheta$ . Due to independence,  $p(v_n | v_1, \dots, v_{n-1}) = p(v_n)$  and therefore  $p(v_1, v_2, \dots, v_n) = \{p(v_1)\}^n$ . Thus,  $H(\mathcal{V}) = H'(\mathcal{V}) = \lg 8 = 3$  bits.

**Order-0 model:**  $V_i$ s are independent and identically distributed with distribution  $p(v) = \pi_v$ ,  $\forall v_i \in \vartheta$ . Due to independence,  $p(v_n | v_1, \dots, v_{n-1}) = p(v_n)$  and therefore  $p(v_1, v_2, \dots, v_n) = \prod_{i=1}^n p(v_i)$ . So,  $H(\mathcal{V}) = H'(\mathcal{V}) = \sum_{v \in \vartheta} p(v) \lg p(v) = \sum_{v_i \in \vartheta} -\pi_v \lg \pi_v = \frac{10}{23} \lg \frac{23}{10} + \frac{8}{23} \lg \frac{23}{8} + \frac{3}{23} \lg \frac{23}{3} + \frac{2}{23} \lg \frac{23}{2} \approx 1.742$  bits.

**Order-1 model:**  $V_i$ s form Markov chain, which means that  $p(v_n | v_1, \dots, v_{n-1}) = p(v_n | v_{n-1}) = P_{v_i, v_j}$ . Substituting  $p(v) = \pi_v$  in Equation (6), we get,

$$H(\mathcal{V}) = -\sum_i \pi_i (\sum_j P_{i,j} \lg P_{i,j})$$

$$= \frac{9}{22} (\frac{2}{3} \lg \frac{3}{2} + \frac{1}{3} \lg 3) + \frac{4}{11} (\frac{3}{8} \lg \frac{8}{3} + \frac{1}{2} \lg 2 + \frac{1}{8} \lg 8) + \frac{3}{22} (3 \times \frac{1}{3} \lg 3) + \frac{1}{11} (2 \times \frac{1}{2} \lg 2) \approx 1.194$$
 bits. The  $\pi_v$  values come from the solution of  $\mathbf{\Pi} \times \mathbf{P} = \mathbf{\Pi}$ .

A few observations are in order. First, at most three bits are sufficient to span the message space of eight alternatives. All three bits of uncertainty exist in order-(-1) model. Thus the model itself is not at all informative, and is the one that maximizes entropy rate. Order-0 and order-1 models show gradual improvement in richness with decreasing entropy rate. Second, the two kinds of entropy rate  $H(S)$  and  $H'(S)$  are equal due to the independence in order-(-1) and order-0 models. The question is whether this is true for models of order-1 or higher. An important result that follows from the earlier definitions clarifies this issue [9].

**Result 1** For any set  $\{V_1, V_2, \dots, V_k\}$  of  $k$  discrete random variables with a joint distribution

$$p(v_1, v_2, \dots, v_k) = \Pr[V_1 = v_1, V_2 = v_2, \dots, V_k = v_k], \quad \forall i(v_i \in \vartheta)$$

the joint entropy  $H(V_1, V_2, \dots, V_k)$  is given by

$$H(V_1, V_2, \dots, V_k) = \sum_{i=1}^k H(V_i | V_1, V_2, \dots, V_{i-1}) \quad (10)$$

Suppose the random variables  $V_i$ s are taken from  $\mathcal{V}$ . Equation (10) reveals an interesting relationship when  $k-1$  is the highest order context under consideration. Substituting  $k=3$  for example, we get

$$H(V_1, V_2, V_3) = H(V_1) + H(V_2 | V_1) + H(V_3 | V_1, V_2).$$

The additive terms on the right-hand side consists of the information carried by levels 1, 2 and 3 (root at level zero) of the trie in Figure 3. Higher order context models are thus more information-rich as compared to the lower order ones. Another way to look at it is that the lower order models mislead the algorithm designer by projecting an under-estimate of uncertainty. To see that the trie holds all the necessary parameters to compute  $H(V_i | V_1, V_2, \dots, V_{i-1})$ , we expand using Equation (6) to find

$$H(V_i | V_1, V_2, \dots, V_{i-1}) = \sum_{\vartheta^{i-1}} p(v_1, \dots, v_{i-1}) \times \left\{ \sum_{\vartheta} p(v_i | v_1, \dots, v_{i-1}) \lg p(v_i | v_1, \dots, v_{i-1}) \right\}. \quad (11)$$

The probabilities  $p(v_1, \dots, v_{i-1})$  and  $p(v_i | v_1, \dots, v_{i-1})$  are estimated from the relative frequencies preserved in the trie. Since the conditional entropy computation for order- $i$  requires the joint distribution for all orders up to  $(i-1)$ , we need to maintain models of all orders up to a suitably large value. Note that order-1 model is an exception due to the simplicity in computing the vector  $\mathbf{\Pi}$  from matrix  $\mathbf{P}$ .

As a part of illustration, let us compute the conditional entropies for contexts of orders 0, 1 and 2 from Figure 3. We have,  $H(V_1) = 1.742$  as before. However,  $H(V_2 | V_1) = \frac{10}{23} (\frac{2}{3} \lg \frac{3}{2} + \frac{1}{3} \lg 3) + \frac{8}{23} (\frac{3}{8} \lg \frac{8}{3} + \frac{1}{2} \lg 2 + \frac{1}{8} \lg 8) + \frac{3}{23} (3 \times \frac{1}{3} \lg 3) + \frac{2}{23} (2 \times \frac{1}{2} \lg 2) \approx 1.182$ . The deviation from the value 1.194 obtained before is due to the fact that we consulted the steady-state distribution from order-0 instead. Similarly,  $H(V_3 | V_1 V_2) = \frac{6}{23} (\frac{1}{3} \lg 3 + \frac{2}{3} \lg \frac{3}{2}) + \frac{3}{23} (3 \times \frac{1}{3} \lg 3) + \frac{3}{23} (\frac{2}{3} \lg \frac{3}{2} + \frac{1}{3} \lg 3) + \frac{4}{23} (\frac{1}{4} \lg 4 + \frac{3}{4} \lg 4) \approx 0.707$ . Finally,  $H(V_1, V_2, V_3) \approx 3.631$  and by taking the running average, we arrive an estimate of  $H(\mathcal{V}) = (1.742 + 1.1818 + 0.707)/3 \approx 1.21$ . As the order increases, this running average estimate should converge to the true value of the per-symbol entropy rate, if it exists.

A natural question arises regarding the highest order  $k$ , which dictates the height of the trie. The answer comes from the following result [9].

**Result 2** For a stationary stochastic process  $\mathcal{V} = \{V_i\}$ , the conditional entropy  $H(V_n | V_1, \dots, V_{n-1})$  is a decreasing function in  $n$  and has a limit  $H'(\mathcal{V})$ .

Clearly, the marginal improvement in model richness starts to die out soon. Intuitively, the largest meaningful order has something to do with largest chain of dependency observed in the movement history. This may come from the longest route taken by the user or the longest stay at a zone (reported by the time-based update only). This becomes evident when we look at the left-hand side of Equation (10). This is the joint entropy of the  $k$ -grams. According to Equation (8), the per-symbol entropy rate would then represent the running average of conditional entropy rates, giving rise to the result [9]:

**Result 3** For a stationary stochastic process  $S$ , both the limits in equations (8-9) exist and are equal, i.e.,

$$H(\mathcal{V}) = H'(\mathcal{V}). \quad (12)$$

For a universal model, all we now need is a way to automatically arrive at the appropriate order dictated by the input sequence. Fortunately, this is exactly what a class of compression algorithms achieves.

## 6 Update and Paging

In Section 2 we reasoned why the time-based and movement-based update schemes are more localized as compared to the

distance-based scheme. Let us recall that this was due to the system's involvement in re-computing and uploading the mobile's new neighborhood after each update. The mobile has to maintain a list of neighboring cells in a cache until the next update. Proponents of the movement-based schemes have also touted a caching scheme such that the new cell-id is cached on every boundary crossing and entry into a cell already in cache is not counted towards reaching threshold  $M$ . This is supposed to avoid some unnecessary updates when cells are re-visited.

Our update scheme essentially uses an enhanced form of caching. Discussions on richness of models in Section 4 reveals the need for gathering statistics based on contexts seen in the movement history. While the zone-ids are treated as character symbols, these contexts must be treated as phrases. A dictionary of such phrases must replace the cache. Whereas just doubling or tripling the threshold only eliminates some primary updates, our algorithm tries to hold them back and send them together in a merged way. Being motivated by the dictionary-based LZ78 compression algorithm originally proposed by Ziv and Lempel [31], it assumes the name "LeZi-update" (pronounced "lazy update").

### 6.1 The LeZi-update algorithm

Before describing the algorithm, let us clarify a very important point: LeZi-update is not meant to replace the threshold-based dynamic update schemes. Rather, it is supposed to reduce the update cost by working as an add-on module to the underlying update scheme. The responsibility of generating the movement history " $v_1v_2v_3\dots$ " still lies with the primary update scheme as before. Let us identify this process of data acquisition as *sampling*. However, a real update message is not generated for each *sampled symbol*. The LeZi-update algorithm captures the sampled message and tries to process it in chunks, thereby delaying the actual update for some sampled symbols. When it finally triggers an actual update, it reports in an encoded form the whole sequence of sampled symbols withheld since the last reporting. In effect, the movement history " $v_1v_2v_3\dots$ " reaches the system as a sequence " $C(w_1)C(w_2)C(w_3)\dots$ ", where  $w_i$ s are non-overlapping segments of the string " $v_1v_2v_3\dots$ " and  $C(w)$  is the encoding for segment  $w$ . The *prime requirement for LeZi-update* (following LZ78) is that *the segments  $w_i$ s must be distinct*.

The system's knowledge about the mobile's location always lags by at most the gap between two updates. The uncertainty increases with this gap, yet larger gaps reduce the number of updates. A natural thing to do would be to delay the update if the current string segment being parsed has been seen earlier, i.e., the path traversed since the last update is a familiar one. Although the gap goes on increasing, it is expected that the information lag will not affect paging much if the system can somehow make use of the profile generated so far. This prefix-matching technique of parsing is the basis of the LZ78 compression algorithm, which encodes variable length string segments using fixed-length dictionary indices, while the dictionary gets continuously updated as new phrases are seen. In figures 4-5, we outline this greedy parsing technique from the classical LZ78, as used in our context. The mobile acts as the *encoder*, while the system takes the role of the *decoder*. It must however not be overlooked that the LeZi-update really makes a paradigm shift from the existing *zone-based* to a new *path-based* update messaging. Re-designing the message format would thus be necessary.

```

initialize dictionary := null
initialize phrase w := null
loop
  wait for next symbol v
  if(w.v in dictionary)
    w := w.v
  else
    encode <index(w),v>
    add w.v to dictionary
    w := v
  endif
forever

```

Figure 4: Encoder at the mobile

```

initialize dictionary := null
loop
  wait for next codeword <i,s>
  decode phrase := dictionary [i].s
  add phrase to dictionary
  increment frequency for every
    prefix of phrase
forever

```

Figure 5: Decoder at the system

### 6.2 Incremental parse tree

The LZ78 algorithm emerged out of a need for finding some *universal variable-to-fixed* coding scheme, where the coding process is interlaced with the learning process for the source characteristics. The key to the learning is a de-correlating process, which works by efficiently creating and looking up an explicit dictionary. The algorithm [31] parses the input string " $v_1, v_2, \dots, v_n$ " ( $v \in \vartheta$ ) into  $c(n)$  distinct substrings  $w_1, w_2, \dots, w_{c(n)}$  such that for all  $j \geq 1$ , the prefix of substring  $w_j$  (i.e. all but the last character of  $w_j$ ) is equal to some  $w_i$ , for  $1 \leq i < j$ . Because of this *prefix property*, substrings parsed so far can be efficiently maintained in a trie [15].

Figure 6 shows the trie formed while parsing the movement history " $aaababbbbbaabccddcbaaaa\dots$ " as " $a, aa, b, ab, bb, bba, abc, c, d, dc, ba, aaa, \dots$ ". Commas separate the parsed phrases and indicate the points of updates. In addition to representing the dictionary, the trie can store statistics for contexts explored, resulting in a *symbol-wise model* for LZ78. A new dictionary entry can only be created by concatenating a single symbol  $v$  to a phrase  $w$  already in it. Thus  $c(n)$  also capture the storage requirement for maintaining the dictionary at both mobile and system side.

It is easy to see that as the process of incremental parsing progresses, larger and larger phrases accumulate in the dictionary. Consequently estimates of conditional probabilities for larger contexts start building up. Intuitively, it would gather the predictability or richness of higher and higher order Markov models. Since, there is a limit to the model richness for stationary processes, the Lempel-Ziv symbol-wise model should eventually converge to the universal model. A result from [12] states that

**Result 4** *The symbol-wise model created by the incremental parsing asymptotically outperforms a Markov model of any finite order and attains the finite-state predictability. At any point, the effective number of states in the incremental*

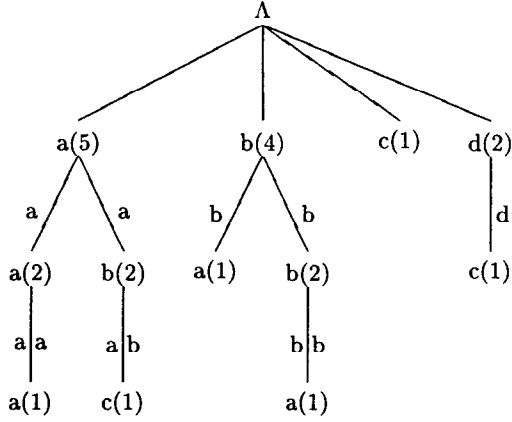


Figure 6: Trie for the classical LZ symbol-wise model

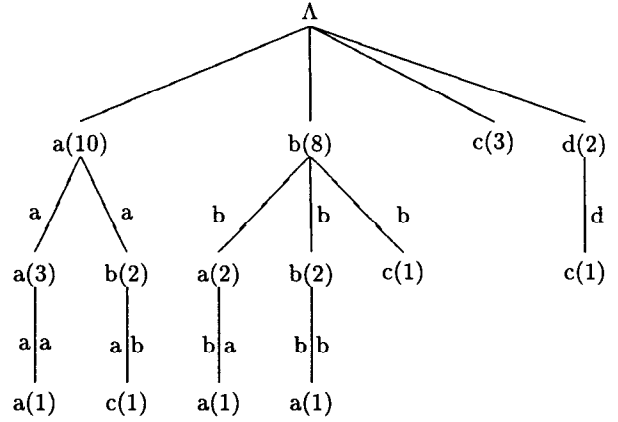


Figure 8: Trie for the enhanced LZ symbol-wise model

parsing model is  $O(c(n))$  and the equivalent Markov order is  $O(\log c(n))$ . Moreover, for stationary ergodic sources, it attains the predictability of the universal model.

For our running example, the largest order we see is 2. However, not all the order-2 contexts get detected. The first reason behind this is that the algorithm remains unaware about the contexts that cross over phrase boundaries. Unfortunately, the algorithm has to work on one phrase at a time. The second reason is that the decoding algorithm, as is, logs the statistics only for the prefixes of the decoded phrases. None of them influence the asymptotic behavior, but the rate of convergence gets affected to a considerable extent. A simple modification on the decoder (system side) as in Figure 7 improves the performance if the dictionary is augmented by the suffixes of the decoded phrases. The enhanced trie for the symbol-wise model is shown in Figure 8. To appreciate the effect, let us compute

```

initialize dictionary := null
loop
  wait for next codeword <i,s>
  decode phrase := dictionary [i].s
  add phrase to dictionary
  increment frequency for every prefix of
    every suffix of phrase
forever

```

Figure 7: Enhanced decoder at the system

the conditional entropies for both the classical and the enhanced symbol-wise models. For the classical one,  $H(V_1) = \frac{5}{12} \lg \frac{12}{5} + \frac{1}{3} \lg 3 + \frac{1}{12} \lg 12 + \frac{1}{6} \lg 6 \approx 1.784$ , and  $H(V_2|V_1) = \frac{5}{12} (2 \times \frac{1}{2} \lg 2) + \frac{1}{3} (\frac{1}{3} \lg 3 + \frac{2}{3} \lg \frac{3}{2}) \approx 0.723$  bits. An estimate for  $\bar{H}(\mathcal{V})$  is  $(1.784 + 0.723)/3 = 0.836$  bits. Conditional entropy for all order-2 contexts turns out to be zero. This is also true for the enhanced one. However, we still have,  $H(V_1) = \frac{10}{23} \lg \frac{23}{10} + \frac{8}{23} \lg \frac{23}{8} + \frac{3}{23} \lg \frac{23}{3} + \frac{2}{23} \lg \frac{23}{2} \approx 1.742$  bits. Then,  $H(V_2|V_1) = \frac{10}{23} (\frac{3}{5} \lg \frac{5}{3} + \frac{2}{5} \lg \frac{5}{2}) + \frac{8}{23} (2 \times \frac{2}{5} \lg \frac{5}{2} + \frac{1}{5} \lg 5) \approx 0.952$  bits. The estimate for  $\bar{H}(\mathcal{V})$  is  $(1.742 + 0.952)/3 = 0.898$  bits, showing that the model has improved to some extent.

### 6.3 Update cost and message complexity

Had the update cost been measured in terms of the volume of update messages in bits, the cost minimization problem can be easily identified with compressing the movement history. However, only the number of updates contributes to the update cost. In an off-line scenario where both the dictionary and the input string are given, it is possible to find out a parsing that minimizes the number of parsed phrases using either a breadth-first search or dynamic programming. But, when the history is generated on-line and the dictionary is built adaptively, we do not have that luxury. The literature has an asymptotic optimality result for Lempel-Ziv parsing in terms of the total volume of update messages, while only an upper bound on the growth rate can be derived for the number of updates [9].

**Result 5** For a stationary ergodic stochastic process  $\mathcal{V} = \{V_i\}$ , if  $l(V_1, V_2, \dots, V_n)$  is the Lempel-Ziv codeword length associated with  $V_1, V_2, \dots, V_n$ , then

$$\limsup_{n \rightarrow \infty} \frac{1}{n} [l(V_1, V_2, \dots, V_n)] = H(\mathcal{V}) \quad (13)$$

with probability 1.

**Result 6** The number  $c(n)$  of phrases in a distinct parsing of a string " $v_1, v_2, \dots, v_n$ " satisfies

$$c(n) = o\left(\frac{n}{\log n - \log \log n}\right) \quad (14)$$

where the base of the logarithms is  $|\mathcal{V}|$ .

Result 5 essentially implies that LZ78 approaches optimality asymptotically for stationary sources. This coveted property is inherited by the LeZi-update, which replaces  $n$  location-updates of the primary update algorithm by  $c(n)$  path-updates. Assuming that the update messages handle the path-updates with no extra cost, the improvement in update cost from Equation (14) is  $\omega(\log n - \log \log n)$ . According to the algorithms in Figure 4-5, the LeZi-update adds only one entry to the dictionary per update. So, the size of the dictionary after  $n$  samplings is also given by  $c(n)$ . The index needed for encoding would have  $O(\lg c(n)) = O(\lg n - \lg \log n)$  bits.

## 6.4 Profile based paging

The location database of every user holds a trie, which is the symbol-wise context model corresponding to the enhanced Lempel-Ziv incremental parse tree. Each node except for the root preserves the relevant statistics that can be used to compute total probabilities of contexts as well as conditional probabilities of the symbols based on a given context. A path from the root to any node represents a context, and the sub-trie rooted at that node reveals the conditional probability model given that context. The paths from the root to the leaves in the Lempel-Ziv trie represent the largest contexts, which are not contained in any other contexts.

Due to space limitation, we informally describe the underlying principles behind the probability assignments. These are essentially similar to those used in *prediction by partial match* (PPM) family of text compression schemes [5, 8]. However, while the PPM techniques are concerned with the probability of the next symbol from the trie, we are interested in the probability of occurrence of specific zone on the probable paths of the mobile, until it sends the next update. These paths are the sequences generated when traversing from the root to the leaves of the sub-trie representing the current context. The desired probability can thus be estimated as the ratio of the number of such paths containing the symbol to the total number of such paths. The estimated conditional probabilities for all the symbols at the current context give rise to the conditional probability distribution given that context. Instead of relying completely on the conditional probability estimates given the context of a specific order, a PPM-style *blending* of these distributions is recommended [5, 8]. A family of paging algorithms comes out based on the choice of these blending strategies.

Let us illustrate the technique by tracing the enhanced Lempel-Ziv trie in Figure 8. Assume that the call has arrived, and no LeZi-style path update message has been received since receiving “aaa” at 9 p.m. Since the path-length of the last update message was three, we can only estimate the probability distribution with orders 0, 1 and 2. For an order-2 estimate we trace the path “aa” from the root of the trie. The sub-trie below this node has only a single child, and hence only one path “a” to a leaf. Based on the occurrence of symbols  $a$ ,  $b$ ,  $c$  and  $d$  on this path, we end up with the assignment of  $\mathbf{p}_2 = [1, 0, 0, 0]$ . Next, we consider the order-1 estimate based on the context “a”. The sub-trie rooted at this node has the following paths, with relative frequencies written in parenthesis:  $a(2)$ ,  $aa(1)$ ,  $b(1)$ ,  $bc(1)$ . Based on the relative frequencies of occurrence, we come to the probability assignment of  $\mathbf{p}_1 = [\frac{1}{2}, \frac{1}{3}, \frac{1}{6}, 0]$ . Finally, for order-0 estimate we start from the root of the whole tree and come to the assignment  $\mathbf{p}_0 = [\frac{2}{5}, \frac{1}{3}, \frac{1}{5}, \frac{1}{15}]$  by proceeding in the same manner. Given a blending weight vector  $w = [w_0, w_1, w_2]$ , the blended probability assignment would turn out to be

$$\mathbf{p} = w_0\mathbf{p}_0 + w_1\mathbf{p}_1 + w_2\mathbf{p}_2. \quad (15)$$

Paging occurs by ordering zones according to a decreasing sequence of probability values from  $\mathbf{p}$ .

Probabilistic prediction is not necessarily the only advantage of building the Lempel-Ziv tries. Search can also be done on the entire trie or a context sub-trie using breadth-first, depth-first or a combined strategy. Geographical information about the neighborhood of LAs come as a by-product of constructing these tries, especially when the movement-based update scheme with  $M = 1$  is used for sampling. A breadth-first search pages the mobile in the neighboring

zones where it is known to have gone before. This is really a pruned version of paging tier by tier [19]. On the other hand, a depth first search can be spawned biased by the weights of the paths emanating from the node and finishing in a leaf, which gives preference to movement in specific directions. An ordering of zones is always prepared for paging, irrespective of the criterion used for this ordering. This list, of course, is guaranteed not to include a zone never visited before, based on the protocol that the mobile will update if the user ends up moving to a new location.

## 7 Conclusions

We have identified the uncertainty due to user mobility as the complexity of the location management problem in a cellular environment. Therefore, entropy is the natural measure for comparing mobility tracking algorithms. As entropy is oftentimes quantified in terms of the number of bits, we arrive at a very intuitive definition of location uncertainty as the least amount of message that needs to be exchanged so that the exact location is known. This provides a common ground for comparing various techniques that are proposed in the literature.

The main difference between our approach and the traditional track of research on location management in cellular PCS network lies in the perspective. We do not formulate mobility tracking as a static optimization problem specified in terms of some known input parameters. The scope of our study extends to the estimation of those parameters from data provided by the underlying sampling mechanism. The update schemes proposed in the literature so far are essentially a characterization of these sampling mechanisms. While sampling is not the main focus of this paper, we have pointed out that one can obtain highly informative samples by combining the time-based and movement-based schemes. The important issue is that the sampled data may have enough redundancy, and need not be sent “as is” in the update messages. Characterizing the user movement data as a stochastic process, we have identified stationarity to be the sufficient criterion for learning the mobility profile. Using entropy as the basis of comparing models, we have illustrated the existence and properties of a universal model when the profile is stationary.

We have adopted the LZ78 compression algorithm as the basis of our update scheme. The choice was guided by two factors, viz. the existence of an explicit symbol-wise context model created by incremental parsing, and the tendency of this model to asymptotically converge to a universal one. As seen in lossless data-compression [5], piecewise stationarity of the sampled sequence should help LZ78 to achieve a high level of performance. The use of pure LZ78 in this paper is motivated by its simplicity in establishing the theoretical basis. Practical implementation would call for a fancier LZ78 variant which must work efficiently with limited memory and should have the capability of forgetting beyond the recent past. We briefly describe how a paging strategy is to be derived out of such a model.

The principle of using compression techniques in mobility prediction may have a bigger potential than just efficient location management. By maintaining global dictionaries along with individual user profile decoded from updates, it may be possible to predict group behavior. In wireless data networks, this can lead to a more efficient bandwidth management and quality of service (QoS) provisioning based on reservation.

**Acknowledgments:** We would like to thank Stephen R. Tate for many helpful suggestions and comments which improved the quality of this paper.

## References

- [1] I. F. Akyildiz and J. S. M. Ho, "Dynamic mobile user location update for wireless PCS networks," *Wireless Networks*, 1(2):187-196, July 1995.
- [2] I. F. Akyildiz and J. S. M. Ho, "Movement-based location update and selective paging for PCS networks," *IEEE/ACM Transactions on Networking*, 4(4):629-638, December 1995.
- [3] A. Bar-Noy and I. Kessler, "Tracking mobile users in wireless communication networks," *IEEE Transactions on Information Theory*, 39(6):1877-1886, November 1993.
- [4] A. Bar-Noy, I. Kessler and M. Sidi, "Mobile users: To update or not to update?," *Wireless Networks*, 1(2):175-185, July 1995.
- [5] T. C. Bell, J. G. Cleary and I. H. Witten, *Text Compression*, Prentice Hall, 1990.
- [6] Y. Birk and Y. Nachman, "Using direction and elapsed-time information to reduce the wireless cost of locating mobile units in cellular networks," *Wireless Networks*, 1(4):403-412, December 1995.
- [7] T. X. Brown and S. Mohan, "Mobility management for personal communication systems," *IEEE Transactions on Vehicular Technology*, 46(2):269-278, May 1997.
- [8] J. G. Cleary and I. H. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, 32(4):396-402, April 1984.
- [9] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley, 1991.
- [10] K. M. Curewitz, P. Krishnan and J. S. Vitter, "Practical prefetching via data compression," *Proc. ACM International Conference on Management of Data (SIGMOD)*, 257-266, May 1993.
- [11] S. K. Das and S. K. Sen, "A new location update strategy for cellular networks and its implementation using a genetic algorithm," *Proc. ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'97)*, 185-194, September 1997.
- [12] M. Feder, N. Merhav and M. Gutman, "Universal prediction of individual sequences," *IEEE Transactions on Information Theory*, 38(4):1258-1270, July 1992.
- [13] J. S. M. Ho and I. F. Akyildiz, "Mobile user location update and paging under delay constraints," *Wireless Networks*, 1(4):413-425, December 1995.
- [14] S. J. Kim and C. Y. Lee, "Modeling and analysis of the dynamic location registration and paging in microcellular systems," *IEEE Transactions on Vehicular Technology*, 45(1):82-90, February 1996.
- [15] G. G. Langdon, "A note on Ziv-Lempel model for compressing individual sequences," *IEEE Transactions on Information Theory*, 29(2):284-287, March 1983.
- [16] G. L. Lyberopoulos, J. G. Markoulidakis, D. V. Polymeros, D. F. Tsirkas and E. D. Sykas, "Intelligent paging strategies for third generation mobile telecommunication systems," *IEEE Transactions on Vehicular Technology*, 44(3):543-553, August 1995.
- [17] U. Madhow, M. L. Honig and K. Steiglitz, "Optimization of wireless resources for personal communications mobility tracking," *IEEE/ACM Transactions on Networking*, 3(6):698-707, December 1995.
- [18] M. Mouly and M.-B. Pautet, *The GSM system for mobile communications*, 1992.
- [19] D. Munoz-Rodriguez, "Cluster paging for traveling subscribers," *Proc. IEEE Vehicular Technology Conference*, 1990.
- [20] D. Plassmann, "Location management strategies for mobile cellular networks of 3rd generation," *Proc. 44th IEEE Vehicular Technology Conference*, 649-653, June 1994.
- [21] G. P. Pollini and C.-L. I, "A profile-based location strategy and its performance," *IEEE Journal on Selected Areas in Communications*, 15(8), October 1997.
- [22] J. Rissanen and G. G. Langdon, "Universal modeling and coding," *IEEE Transactions on Information Theory*, 27(1):12-23, January 1981.
- [23] J. Rissanen, "A Universal data compression system," *IEEE Transactions on Information Theory*, 29(5):656-664, September 1983.
- [24] C. Rose and R. Yates, "Minimizing the average cost of paging under delay constraints," *Wireless Networks*, 1(2):211-219, July 1995.
- [25] C. Rose, "Minimizing the average cost of paging and registration: A timer-based method," *Wireless Networks*, 2(2):109-116, June 1996.
- [26] C. E. Shannon, "The mathematical theory of communication," *Bell System Technical Journal*, 27:379-423, 623-659, 1948, reprinted in *The Mathematical Theory of Communications* by C. E. Shannon and W. Weaver, University of Illinois Press, 29-125, 1949.
- [27] J. S. Vitter and P. Krishnan, "Optimal prefetching via data compression," *IEEE Symposium on Foundation of Computer Science*, October 1991 or *Technical report BROWN-CS-91-46* (extended version).
- [28] M. J. Weinberger, J. J. Rissanen and M. Feder, "A universal finite memory source," *IEEE Transactions on Information Theory*, 41(3):643-652, May 1995.
- [29] T. A. Welch, "A technique for high-performance data compression," *IEEE Computer*, 17(6):8-19, June 1984.
- [30] H. Xie, S. Tabbane and D. Goodman, "Dynamic location area management and performance analysis," *Proc. 43rd IEEE Vehicular Technology Conference*, 533-539, May 1993.
- [31] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, 24(5):530-536, September 1978.
- [32] M. M. Zonoozi and P. Dassanayake, "User mobility modeling and characterization of mobility patterns," *IEEE Journal on Selected Areas in Communications*, 15(7):1239-1252, September 1997.