LATS: A Load-Adaptive Threshold Scheme for Tracking Mobile Users

Zohar Naor, *Student Member, IEEE,* and Hanoch Levy, *Member, IEEE*

***Abstract*— Mobile user tracking is a major issue in wireless networks. Previous studies and traditional approaches dealt only with tracking algorithms which adapt themselves to the user activity. In this work, we propose a novel approach for user tracking, in which the tracking activity is adapted to both user and system activity. The basic idea is to make the user-location update-rate dependent not only on the user activity (such as the call profile and mobility pattern). Rather, it is also made dependent on the signaling load, which reflects the actual cost of the update operation. Thus, at low-signaling load locations, the users are to transmit location update messages more frequently. To carry out this approach, we propose a load-adaptive threshold scheme (LATS): the network determines for each cell a *registration threshold level* (which depends on the cell load) and announces it, as a broadcast message, to the users. The user computes its own *registration priority* and then transmits a registration message only if its priority exceeds the announced threshold level. Thus, whenever the local load on the cell is low, the registration activity increases, while in loaded cells the registration activity decreases. Our analysis shows that the LATS reduces the paging cost, in comparison with other dynamic methods, without increasing the wireless cost of registration. Moreover, if higher user density is coupled with less mobility (e.g., consider vehicles), then the LATS strategy offers further performance improvement. The load-adaptive strategy can be used *in addition* to any other dynamic tracking strategy. Furthermore, the computational complexity imposed on the user is identical to that required by an equivalent load-insensitive scheme.**

***Index Terms*—Mobile, PCS, user tracking, wireless.**

I. Introduction

THE GROWING demand for personal communication services (PCS) increases the need for efficient utilization of the limited radio resources available for wireless communication. In this work, our concern is in the utilization of the wireless resources devoted to location management. The problem addressed in this paper is the minimization of the wireless cost of mobile user tracking in PCS networks.

The utilization of wireless network resources for mobile user tracking has been addressed by many studies. Basically, there are two extreme strategies that may be used for user tracking. In the first strategy, known as "Never Update," the user never updates its location. Thus, whenever there is a need to set up an incoming call directed to the user, the system must search for the user all over the network. In the other extreme strategy, known as "Always Update," the network continuously keeps track of the user location. Various strategies which combine these two extremes have been proposed in the literature [1]–[3], [6]–[9].

The basic idea shared by these papers is known as the "Partial Registration" strategy. Namely, upon location change, the user may or may not update its new location. The criterion for user registration may be static, such as network partition into *location areas*. For example, in the Global System for Mobile Communications (GSM), the network is partitioned into groups of cells, referred to as *location areas*. The user updates its location each time it changes a location area, while within a location area it uses the "Never Update" strategy. Since the partition into location areas is static and done by the system, not accounting for the user-dynamic behavior, we consider this strategy as static. Another type of partial registration strategy is the dynamic strategy, in which each user decides when and where to update its location. The criterion for user registration may be a function of time [8], distance from last known location [2], [6], number of movements between cells [2], or based on personal location profile [9]. All the dynamic partial registration strategies mentioned above are implemented solely on user equipment. As such, they ignore the system activity, and depend solely on the user activity.

The optimal solution which minimizes the user tracking cost using these methods is of high computational complexity, and often requires a dynamic programming method. Some of these strategies (such as the distance-based strategy) require information that is not generally available to the user. Thus, an implementation of an optimal solution on the user equipment is not feasible, due to commercial, maintenance and reliability reasons. In practice, the users may register using a fixed, predefined parameter (timer, distance, etc.), disregarding the exact details of the user activity. Clearly, the performance of such an implementation is inferior to the original strategy. In addition, since the user decision whether to register (update) or not to register ignores the status of other users, the likelihood of collision is expected to be very high, especially at high load periods.

Recognizing these drawbacks, we propose to shift a significant part of the tracking activity from the user equipment to the system equipment, and to integrate more intelligence on the network side. The basic idea is to leave user-specific decisions in the user's equipment while moving network general decisions to the network. Guided by this basic idea, we propose a novel approach for mobile user tracking, in which the user registration is based not only on its own

Manuscript received April 30, 1989; revised December 11, 1998; approved by IEEE/ACM Transactions on Networking Editor S. Chanson.

The authors are with the Department of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel (e-mail: zohar@math.tau.ac.il).

Publisher Item Identifier S 1063-6692(99)09852-0.

activity, but also on local network load and the status of other users within the same cell. The implementation of this approach is achieved via the following mechanism. The user computes its *registration priority* based on its own activity. On the other hand, the network determines, for each cell, a *registration threshold level*, based on cell load, time of day, day of week, etc. This parameter, which is unique for each cell, is transmitted by each base station as a broadcast message to all the users within the cell through the down-link control channel. Finally, the decision of when to transmit a registration message is done by the user, but is based on both parameters. Such a message is sent whenever the user *registration priority* exceeds the cell *registration threshold level*.

The advantage of the proposed method on other methods is in taking into account the system activity. Less critical registration messages are avoided during heavy traffic periods, while low traffic periods and areas are used to gather extensive information on the user location.

Our analysis shows that the load-adaptive threshold scheme (LATS) strategy offers a significant improvement not only at lowly loaded (LL) cells, as someone would intuitively expect, but also at highly loaded cells. This is done by eliminating unnecessary searches after users residing in LL cells. Moreover, if higher user density is coupled with less mobility, e.g., as in vehicular motion, the LATS strategy offers further performance improvement.

The concept of an *adaptive threshold* scheme is not limited to the problem of tracking mobile users, and can be used in a much wider range of applications in the area of shared media networks. Load-sensitive algorithms, in many areas, are implemented either by a central approach, in which system resources are allocated to the users by the system (e.g., polling systems), or by a distributed approach, in which each user makes its own decisions using an agreed-upon MAC protocol. The *adaptive threshold* approach suggests another way. Due to the huge number of users in a PCS network, the media access algorithm (i.e. the registration) must be distributed, where each user makes its own decision when and where to update its location. On the other hand, to improve system performance, the network informs the users about the cost of registration. Another load-sensitive approach for tracking mobile users is described in [5], where idle periods are used by the network to actively search for the location of mobile users.

This paper focuses on two issues.

1) Design a registration algorithm, based on a *dynamic threshold*, which is sensitive both to the user activity and to the system (cell) load.
2) Propose an efficient paging algorithm which takes advantage of the dynamic thresholds used in the registration algorithm. The paging algorithm is based on an efficient search in a discrete time–space mobility graph, which enables the system to compute all feasible locations at paging time.

These issues are covered in Section III (after Section II describes the model). Further, in Section IV, we compare the LATS strategy to its corresponding load-insensitive (LI) strategy. The performance of the timer-based LATS is evaluated and compared to the performance of the IS-41 standard [10] under two representative models: 1) a one-dimensional (1-D) mesh topology and a two-dimensional (2-D) mesh topology, both under Gaussian-like shaped load distributions and 2) a 2-D Manhattan-like system, which can model a vehicular motion within an urban area. The results show that the timer-based LATS is superior to its corresponding LI strategy. For a Gaussian-like shaped load distribution, the timer-based LATS has the potential of reducing the paging cost to about half in a 1-D system, and up to four times for a 2-D system.

A summary and concluding remarks are given in Section V.

II. Model and Notation

We consider a wireless network partitioned into cells. To a good approximation, the number of users may be considered infinite, for traffic considerations. The user location is understood as an identifier of the cell in which the user is currently residing. Two cells are called neighboring cells if a user can move from one to the other without crossing any other cell. To model user movement in the network, we assume that time is slotted, and that a user can make, at most, one cell transition during a slot. It is assumed that the movement of the user is done just at the beginning of time slot, such that it precedes any other event, such as a *paging* event. The movements are assumed to be stochastic and independent from one user to another. We assume that calls are initiated by the users as a Poisson process at average-rate λ. The *user roaming interval* is defined in [8] as the time interval since the last contact of the user with the system, and the next paging event to this user.

III. The Tracking Strategy

The tracking algorithm is split between the user and the network. The registration algorithm performed by the user is based on the user activity and the *load factor*, computed by the network. Each cell computes its own *load factor*, independent of the other cells, and announces it to the users as a broadcast message, through the down-link control channel. In addition, each user computes its own registration priority, based on its call and mobility parameters, and computes the local *registration threshold*, based on the local *load factor*. Whenever the user-registration priority exceeds the local registration threshold, the user transmits a location update message. Since the local *registration threshold* depends on the current load on the cell, different cells may have different registration thresholds.

The search algorithm at paging event is restricted only to the set of all feasible user locations. These are all the cells that satisfy two conditions.

1) Their registration threshold at the paging event is higher than the user registration priority.
2) They are reachable from the user last known location during its roaming interval, without sending a registration message on the way.

A. The Registration Strategy

The user registration strategy is to be used *in addition* to another dynamic registration strategy, such as the timer-based method [8], the distance-based method ([2], [6]), or the movement-based method [2]. The basic idea is the following. The user tracks its *registration priority* R_u. For example, under the timer-based method, R_u is the user roaming interval τ. Under the distance- and the movement-based methods, R_u is the distance traveled by the user, and the number of cell transitions since its last location update, respectively. The value of R_u is compared to the *local registration threshold* R_l, given by

$$R_l = \alpha R. \tag{1}$$

The parameter R in (1) is the LI registration threshold, introduced in previous studies. For example, under the timer-based method R is timer T, specified in [8]. The parameter α in (1) is the local *load factor*, received by the user through the down-link control channel. The parameter α is transmitted by each base station as a broadcast message through its down-link control channel (for example, DCCH in the GSM system). It reflects the current local load on the control channel at that cell, relatively to a pre-defined load level. The user transmits a location update message whenever its *registration priority* R_u exceeds the *local registration threshold* R_l

$$R_u \geq R_l = \alpha R. \tag{2}$$

For example, under the timer-LATS method, the user registers whenever its *roaming interval* τ satisfies: $\tau \geq T_l = T\alpha$. Similarly, the distance-LATS *registration threshold* D_l is evaluated as $D_l = D\alpha$, where D is the LI distance threshold specified in [2], [6], and the movement-LATS *registration threshold* M_l is given by $M_l = M\alpha$, where M is the LI movement threshold specified in [2]. Since the local load factor α is transmitted by the base station, the computational complexity imposed on the user is identical to that required by the corresponding LI scheme. For practical reasons, the *local registration threshold* R_l should be bounded from below as to avoid redundant registrations at very lightly loaded areas.

B. Network Algorithm—Cell Level

The *load factor* α is computed for each cell, independently of the other cells, and transmitted by the base station (BS), as a broadcast message, through the down-link control channel. Since registration messages are transmitted through the up-link control channel (UCC), our main concern is to guarantee that an increase in the registration activity will not jam the UCC.

The *load factor* is estimated from the statistical usage of the UCC. The basic idea is the following: let ρ_{UCC} be the UCC utilization, $0 \leq \rho_{\text{UCC}} \leq 1$. Whenever ρ_{UCC} drops below a pre-defined threshold, say ρ_L, the *registration threshold* α decreases. As a result, the registration activity increases at lightly loaded cells. On the other hand, whenever ρ_{UCC} exceeds (another) pre-defined threshold, say ρ_H $(\rho_H > \rho_L)$, the *registration threshold* increases. To stabilize the algorithm, the modification rate of the *registration threshold* follows a Hysteresis curve.

1) The Load-Factor Modification Algorithm: Our goal is to keep the value of ρ_{UCC} in the range $\rho_L \leq \rho_{\text{UCC}} \leq \rho_H$, where ρ_L, ρ_H are pre-defined values, satisfying $0 < \rho_L < \rho_H < 1$. The target UCC utilization is defined as $\rho_O = (\rho_L + \rho_H)/2$. The modification of the load factor α is as follows. Whenever ρ_{UCC} satisfies $\rho_L \leq \rho_{\text{UCC}} \leq \rho_H$, α remains unchanged. If either $\rho_{\text{UCC}} < \rho_L$ or $\rho_{\text{UCC}} > \rho_H$, the new value of α, denoted by α_n, is given by

$$\alpha_n = \alpha_l \frac{\rho_{\text{UCC}}}{\rho_O} \tag{3}$$

where α_l is the last value of α, under which the UCC utilization is ρ_{UCC}. Since $\rho_H > \rho_O > \rho_L$, whenever $\rho_{\text{UCC}} > \rho_H$, we get that $\alpha_n > \alpha_l$. Consequently, the *local registration threshold* $R_l = R\alpha$ increases, and the registration activity decreases. Similarly, if $\rho_{\text{UCC}} < \rho_L$ the registration activity increases.

The modification mechanism given in (3) is based on the assumption that the registration activity linearly depends on the *registration threshold*. This assumption holds for the timer-based method, but does not necessarily hold for other tracking strategies, such as the distance-based method and the movement-based method. Moreover, since the UCC is used for call handling as well as for location management, the *registration threshold* may have to be sensitive to the call traffic as well. For example, if all lines in a certain cell are busy, the registration messages at that cell should get higher priority than requests for call setup. As a result, *the registration activity may increase, even at heavily loaded (HL) cells*. The conclusion from this discussion is that the modification mechanism suggested in (3) may need second-order refinement, taking into account mobility and call parameters. The dependency of ρ_{UCC} on the *registration threshold* can be estimated using standard procedures, such as look-up tables based on network history.

C. The Paging Algorithm

Consider a paging event at time $t = \tau$, for a user u with *roaming interval* equal to τ, whose last known location, at time $t = 0$ is X_0. The *distance* between two cells, say x and y, denoted by $d(x, y)$, is defined as the length of the shortest path between x and y, measured in number of cells. That implies that $d(x, x) = 0$, and that $d(x, y) = 1$ if, and only if, x is a nearest neighbor of y. Our goal is to minimize the number of locations at which the user is paged.

Definition: The *Mobility graph* G is defined as a directed graph, in which all vertices are of the form (x, t) where x is a cell in the network, and t is a time slot. A (directed) edge exists between two vertices, say (x_1, t_1) and (x_2, t_2), in the *Mobility graph* if, and only if: 1) $d(x_1, x_2) \leq 1$ and 2) $t_2 = t_1 + 1$. There exist two different types of edges in the *Mobility graph*.

- A *static* edge is a directed edge from the vertex (x, t) to the vertex $(x, t+1)$, reflecting a situation where the user remains at its last location.
- A *dynamic* edge is a directed edge from the vertex (x_1, t) to the vertex $(x_2, t+1)$, where x_1 and x_2 are nearest neighbors, reflecting a user movement from cell x_1 into cell x_2, at time $t+1$.

Definition: A *nonreporting* (NR) vertex is a vertex (x, t) in the mobility graph that is accessible from $(X_0, 0)$, and its *local* *registration threshold* R_l is higher than the user registration priority.

For example, under the timer-based method, an NR vertex must satisfy the conditions

$$T_l = \alpha(x,t)T > t \tag{4}$$

where T_l is the local timer at cell x at time t, and

$$0 \le d(X_0, x) \le t \le \tau. \tag{5}$$

Equation (5) follows from the assumption that the user can make, at most, one cell transition during a single time slot. This assumption holds if the size of the time slot is chosen such that the user cannot cross more than one cell during a single time slot.

Under the distance-based strategy, let D_l denote the local distance threshold. An NR vertex is defined as a vertex (x,t) which satisfies, in addition to (5), the condition

$$0 \le d(X_0, x) \le D_l = D\alpha(x,t). \tag{6}$$

Similarly, let M_l denote the local registration threshold under the movement-based method. An NR vertex must satisfy both (5) and the condition

$$0 \le d(X_0, x) \le M_l = M\alpha(x,t). \tag{7}$$

An NR vertex describes a point (in time and space), such that at time t, the cell x is reachable from the user last known location, and its *registration threshold* is higher than the user *registration priority*. Hence, an NR vertex depends on the user mobility, as well as on the system connectivity and load distribution.

A *feasible roaming path* is defined as a directed path in the mobility graph, which starts at $(X_0, 0)$, and all the vertices in the path are NR vertices. Given that the user *roaming interval* is equal to τ, its actual roaming path must be a *feasible roaming path* in length τ, starting at $(X_0, 0)$ and terminating at the user location at time τ.

Definition: The user *personal location area* (PLA) at time τ is defined as the set of all cells x, such that the vertex (x,τ) in the mobility graph is an NR vertex connected to the vertex $(X_0, 0)$ through a *feasible roaming path*.

From this definition it follows directly that the user PLA is the group of all its feasible locations at the time of paging. Hence, the user PLA is the minimal set of cells, at which paging the user guarantees a success.

The paging algorithm considered in this study is subject to one-phase paging delay: when a paging event occurs, the user is paged simultaneously at all the cells belonging to the PLA.

1) Constructing the PLA: The first step of the paging algorithm is to find all cells belonging to the user PLA. The algorithm uses a Breadth First Search (BFS) on the mobility graph, beginning from the vertex $(X_0, 0)$, which represents the user last known location X_0 at time $t = 0$. The BFS is conducted in steps, where the k-th step accepts as an input a set $A(k-1)$, produced by the $(k-1)$st step, and generates the set $A(k)$. $A(k)$ is the set of NR vertices of the form (x,k), connected to the vertex $(X_0, 0)$ through a *feasible roaming path*. For each member in $A(k-1)$, say $(x, k-1)$, we consider in the k-th step all vertices of the form (x', k), where x' is either x or a nearest neighbor of x. If (x', k) is an NR vertex, it is added to the set $A(k)$. In the first step, $A(0)$ contains the vertex $(X_0, 0)$, and we consider the nearest neighbors of X_0 at time slot $t = 1$. The search is terminated when either $k = \tau + 1$ or the set $A(k)$ is empty.

Proposition 3.1: The set $A(\tau)$ is the user PLA.

Proof: The proposition can be proved by induction on k that shows that $A(k)$ is the set of all NR vertices of the form (x,k) connected to $(X_0, 0)$ through a feasible roaming path in length k. Thus, $A(\tau)$ is the set of all NR vertices of the form (x,τ) connected to $(X_0, 0)$ through a feasible roaming path. □

Remark 3.1: To construct the PLA for all users, the system keeps, for each cell, a table of its local registration threshold values for each time slot. In addition, the mobility graph is based on the adjacency graph of the network.

The Complexity of the PLA Construction Algorithm: since the BFS is conducted on both time and space, the nearest neighbors of X_0 can be visited in the worst case τ times, while the cells in a distance τ from X_0 are visited at most one time. Recalling that the user can make at most one cell transition during a time slot, a distance (in terms of cells) can be compared to time. Let $n(X_0, j)$ be the number of cells within a distance less or equal to j from X_0, where $n(X_0, 0) = 1$. In the i-th iteration of the BFS algorithm, at most $n(X_0, i)$ vertices are considered. An upper bound on the worst-case complexity of the BFS algorithm is given by $S(X_0, \tau) = \sum_{i=0}^{\tau} n(X_0, i)$. In reality, the number of nearest neighbors of a cell is bounded by some constant, and so is the cell density. Thus, the maximal number of cells reachable during t time-slots is $O(t^2)$, and the worst-case complexity of the BFS algorithm is therefore $S(X_0, \tau) = \sum_{i=0}^{\tau} O(i^2) = O(\tau^3)$. The explanation of the computational complexity is that the BFS on the mobility graph searches in $O(\tau^2)$ cells, each cell has at most τ vertices, yielding $O(\tau^3)$ vertices.

Lemma 3.1: If the number of cells reachable within t time slots is $O(t^2)$, then the construction of the PLA has a worst-case complexity of $\Omega(t^3)$.

The proof is given in Appendix A. The conclusion from Lemma 3.1 is that the computational complexity of the PLA construction algorithm described above cannot be further improved.

Remark 3.2: Note that the process which finds all members of the PLA is a computational task. As such, it makes no use of wireless network resources. The search is done on the mobility graph, not on the network's physical infrastructure.

The computational task required to find the PLA can be significantly reduced by considering all NR vertices of the form (x,τ). Clearly, this group contains the user PLA. For example, under the timer-based LATS, this is simply the group of all the cells at a distance less or equal to τ from x, having local timer $T_l > \tau$. However, the reduction in the computational cost is achieved on the expense of increasing the cost of paging. In practice, the load distribution over the network changes gradually, relatively to the user motion. Thus, the registration threshold is expected to change only every k time slots, where k is a constant. The number of iterations required can be reduced to $x = [\tau/k] + 1$, where in each iteration the BFS is extended by a distance k. The worst-case complexity under this assumption is $O(k^2x^3) \propto \frac{\tau^3}{k}$. This is still $O(\tau^3)$, but the computational task is reduced by a factor k.

D. Tracking Cost Considerations

Below we evaluate the computational cost and wireless bandwidth imposed by the LATS strategy.

The user must track its registration priority, to be compared to the local registration threshold at its current location. The LI registration threshold may be calculated either dynamically, as in [8], [3], and [6], or statically, as in IS-41 standard [10]. Using LI dynamic registration threshold, the *load factor* transmitted by the BS describes the actual cost of registration. Hence, the computation of the local registration threshold is identical to the evaluation of the LI registration threshold described in the above-mentioned studies. The only difference is that the registration cost, which considered as a pre-defined parameter in these studies, is transmitted by the base stations. The computational task imposed on the user is therefore identical to that required by the corresponding LI strategy.

Using a static LI registration threshold, as used for example, in the IS-41 standard, the *local registration threshold* can be determined by a pre-defined look-up table. In this case, the load factor determines the *local registration threshold* (e.g., timer, distance), to be selected from a pre-defined table. This table contains several registration threshold values, for different load conditions.

The wireless bandwidth imposed by the LATS is negligible under both dynamic and static registration threshold. In order to support M different values of the *registration threshold,* only $\lg_2 M$ bits suffice. For example, using timer-based LATS with the IS-41 standard, only two bits are required to support four different timers.

IV. Performance Analysis

In this section, we show that the LATS strategy is superior to its corresponding LI strategy. In addition, the performance of the timer-based LATS is evaluated and compared to that of the timer-based method for two models of load distribution and network connectivity.

Theorem 4.1: Given an LI tracking strategy, its corresponding LATS can only perform better.

Proof: Consider an LI tracking strategy, with registration threshold R. R can be either a timer, a distance, the number of cell transitions, etc. We define a corresponding load-sensitive scheme which is the following special case of LATS. The *load factor* is $\alpha' = \min\{\alpha, 1\}$ where α is the *load factor* given in (1). The load-sensitive *registration threshold* R_l is given by $R_l = R\alpha' \leq R$. Consequently, the registration activity can only increase under the LATS strategy, due to the use of nonutilized system resources, *at practically no cost.* Hence, using the same *actual* bandwidth for registration, the paging cost under LATS can only be lower than that under its corresponding LI scheme. □

In the following sections, the performance of the timer-based LATS is evaluated and then compared to the performance of the timer-based method, as adopted by the IS-41 standard [10]. In Section IV-A we consider a 1-D mesh topology and a 2-D mesh topology, both under Gaussian-like shaped load distributions. In Section IV-B, we consider a Manhattan-like topology system having two load levels. A Gaussian-like shaped load distribution can model a city, while a Manhattan-like network connectivity can model a vehicular motion. In both models, we consider a paging algorithm that is conducted simultaneously at all the cells within the user PLA.

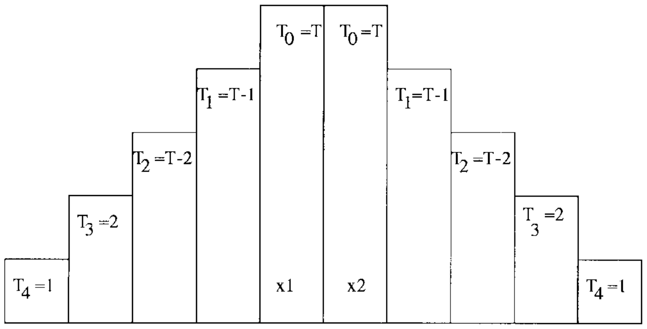


Fig. 1. A Gaussian-like shaped load distribution for a 1-D system, with static timer $T = 5$ time units. The local timer T_i at each cell decreases with the distance of the cell from the system center (cells $x1$ and $x2$).

A. Analysis of a Gaussian-Like Shaped Load Distribution

In this section, the performance of the timer-based LATS is evaluated and then compared to the timer-based method [10], under a Gaussian-like shaped load distribution, in which the load reaches its peak at the network center and decreases gradually toward its periphery.

First, we consider a 1-D system consisting of $2L$ equal-size cells. A model of the system is depicted in Fig. 1 for $L = 5$. The most loaded cells are at the system center (cells $x1$ and $x2$ in Fig. 1). The distance of a cell from the system center is defined to be the minimum of the distances from $x1$ and $x2$. It is assumed that the load decreases linearly with the distance from the system center, that is the load at a distance i from the system center is given by $l_0\frac{L-i}{L}$ $(i = 0, 1, 2 \cdots L-1)$, where l_0 is the load at the system center. Since the load is proportional to the user density, the steady-state probability to find the user at a distance i from the system center $(i = 0, 1, 2 \cdots L-1)$ is given for each side (either left or right) by

$$\pi_i = \frac{L-i}{L(L+1)}. \tag{8}$$

Note that $2\sum_{i=0}^{L-1} \pi_i = 1$. Let T be the static timer [10]. Under the timer-based LATS T is the timer used at the most loaded cells, $x1$ and $x2$. Let T_i be the timer used under LATS at a distance i from the system center. Since the load-sensitive timer is proportional to the user density, (8) suggests that $T_i = T - i$. Since the local timer T_i is bounded from below by 1 ($T_i = 1$ implies a registration message transmitted per every time slot), we get that

$$T_i = \mathrm{maximum}\{1, \mathrm{T - i}\}, \quad \mathrm{i = 0, 1, 2, \cdots, L-1}. \tag{9}$$

The load factor affects the search cost in two ways: first, by reducing the effective timer in the system, and second, by reducing the uncertainty in user location for the same *roaming interval*. As opposed to the timer-based method [10], under which the uncertainty in user location increases with its *roaming interval*, the user PLA actually *decreases* for large *roaming intervals* under the timer-based LATS, since the uncertainty in user location is then restricted to HL cells. For example, consider Fig. 1, and note that if the user *roaming interval* τ equals $T-1$, the only feasible locations are $x1$ and $x2$. For $\tau = T-2$, the PLA size is at most four cells: $x1$, $x2$, and their nearest neighbors. In general, the *exact* size of the PLA depends also on the user last known location. However, in any case, the number of feasible locations is bounded from above by the number of cells having a local timer $T_i > \tau$. Hence, the larger is τ, the smaller is the upper bound on the PLA size. On the other hand, since the user can make at most one movement during a single time slot, for small values of τ, the PLA size is bounded from above by $2\tau+1$. From the above discussion, we get that the upper bound on the PLA size is given by

$$|\mathrm{PLA}(\tau)| \le \mathrm{minimum}\{(2\tau+1), 2(\mathrm{T}-\tau)\}. \tag{10}$$

Hence

$$|\mathrm{PLA}(\tau)| \le 2\tau+1 \tag{11}$$

for $0 \le \tau < \lceil T/2 \rceil$ and

$$|\mathrm{PLA}(\tau)| \le 2(T-\tau), \tag{12}$$

for $T > \tau \ge \lceil T/2 \rceil$. The upper bound on the PLA size is achieved for $\tau = \lfloor T/2 \rfloor$. Using (10)–(12), we get that

$$|\mathrm{PLA}(\tau)| \le T, \quad \forall \tau. \tag{13}$$

For the LI timer-based method, the PLA size is $2\tau+1$, with maximum value equals $2T+1$ achieved at $\tau = T-1$. Hence, the maximal number of searches under the timer-based LATS is reduced, by a factor of at least two, in comparison to the LI timer-based method. Equation (10) implies that the superiority of the timer-based LATS over its corresponding LI scheme increases with τ.

Our objective next is to derive an upper bound on the *expected* search cost under LATS. Using (9), we get that for sufficiently large systems, the condition $T \le L$ implies that $T_i = 1$ for $i \ge T-1$. Hence, using (10)–(12), the expected search cost under the timer-based LATS decreases with L and reaches its maximum in the range $T \le L$ at the point $T = L$. Hence, for sufficiently large systems, the expected search cost is bounded from above by that under the condition $T = L$. Assuming that the call-arrival rate is significantly smaller than $\frac{1}{T}$ (which is true for actual systems), the probability to find the user at the time of paging at a cell with local timer T_i is equal to π_i. Since calls are initiated as a Poisson process which is independent of the user motion, the probability that the user *roaming interval* at the time of paging is equal to τ $(\tau = 0, 1, 2 \cdots T_i - 1)$ is uniformly distributed over $[0, T_i - 1]$. Hence, substitute $T_i = T - i$, $L = T$, and using (8)–(10), an upper bound on the expected number of cells needed to be searched is given by

$$E[\mathrm{Search}_{\mathrm{LATS}}] \le \sum_{i=0}^{T-1} \pi_i \frac{1}{T_i} \sum_{\tau=0}^{T_i-1} \min\{(2\tau+1), 2(T-\tau)\} \tag{14}$$

which yields

$$E[\mathrm{search}_{\mathrm{LATS}}] \le \frac{2}{T(T+1)} \sum_{i=0}^{T-1} \sum_{\tau=0}^{T-i-1} \min\{(2\tau+1), 2(T-\tau)\}. \tag{15}$$

In Appendix B, we show that the value of the expression in (15) is bounded from above by $\frac{T+1}{2}$. Hence, we get that

$$E[\mathrm{search}_{\mathrm{LATS}}] < \frac{T+1}{2} \approx \frac{T}{2}. \tag{16}$$

Under the LI timer-based method [10], the expected search cost is given by

$$E[\mathrm{search}_{\mathrm{LI}}] = \frac{1}{T} \sum_{\tau=0}^{T-1} (2\tau+1) = T. \tag{17}$$

Hence, for sufficiently large networks, the expected search cost under the timer-based LATS is reduced by about half, in comparison to the LI timer-based method.

It is interesting to note that while the expected search cost under the timer-based LATS is about $1/2$ of that under the LI timer-based method, the expected timer used by the user under LATS is as large as roughly $2/3$ of that under the LI timer-based method. To prove it, let T_u be the timer used by the user. Recalling that T_i is the timer used at distance i from the system center, the expected value of T_u under the condition $T = L$ is

$$E[T_u \mid T = L] = 2\sum_{i=0}^{L-1} \pi_i T_i = \frac{2}{T(T+1)} \sum_{i=0}^{T-1} T_i^2 = \frac{2T+1}{3}. \tag{18}$$

We therefore observe that $E[T_u | T = L] > \frac{T}{2}$. Hence, the reduction in the paging cost results not only from effective timer reduction, but also from the fact that under the timer-based LATS, for large roaming intervals the uncertainty in user location is confined to the highly loaded cells.

The technique used for the 1-D model can be easily applied to a similar 2-D model, under which the user can move to each of its eight nearest neighbors. The timer used within the most loaded cell is T. Its eight nearest neighbors use a timer $T_1 = T-1$, their nearest neighbors use a timer $T_2 = T-2$, and so on. Under the LI timer-based method the PLA size is given by $|\mathrm{PLA}(\tau)| = (2\tau+1)^2$, with maximum value equals to $(2T-1)^2$ for $\tau = T-1$. Since the call arrivals are independent of the user motion, the expected search cost under the LI timer-based method is given by

$$E[\mathrm{search}_{\mathrm{LI}}] = \frac{1}{T} \sum_{\tau=0}^{T-1} (2\tau+1)^2 = \frac{4T^2-1}{3}. \tag{19}$$

Under the timer-based LATS, we get that when τ approaches T, the user PLA is restricted to the nearest neighbors of the most loaded cell. Hence, we get that

$$|\text{PLA}(\tau)| \leq \text{minimum}\{(2\tau+1)^2, (2(T-1-\tau)+1)^2\} \tag{20}$$

with an upper bound for $\tau = \lfloor T/2 \rfloor$. The upper bound equals T^2 for T odd, and equals $(T-1)^2$ for T even, which is roughly four-times smaller than the upper bound under the LI timer-based method. A similar analysis to that used in Appendix B can be used for the expected cost analysis.

Remark 4.1: The analysis in this section provides an upper bound on the expected paging cost under the timer-based LATS. In order to obtain the exact number of searches, the user's last known location must be considered. However, due to the dependency of the registration process under LATS on user location, the probability that the user last known location is x is not necessarily the steady state probability of being at location x.

B. Analysis of a Manhattan-Like Topology System (Vehicular Motion)

To demonstrate the timer-LATS performance for vehicular motion we consider a 2-D Manhattan-like system, which is depicted in Fig. 2. The system can model a city with major streets running east–west and north–south. The network connectivity can model a vehicular motion along the streets. The user can move along either the horizontal or the vertical direction. All the cells in the system are assumed to have the same length and width. The cells on the intersections are assumed to be HL, while all other cells are assumed to be LL. Each HL cell is followed by an LL section, consisting of W cells, and terminated by another HL cell. The sequence of an LL section followed by an HL cell repeats periodically along both dimensions, and is referred to as a *block*. From each junction (an HL cell), the user can either turn to the right, turn to the left, or continue in the same direction. Since a high density of vehicles is usually coupled with low mobility, it is assumed that the time period t_h required to cross an HL cell is longer than the time period t_l required to cross an LL cell: $t_l < t_h$.

1) Evaluation of Paging Cost: Let T^l be the timer within the LL cells, and T^h be the timer within the HL cells. Clearly $T^l < T^h$. The distance traveled by the user, in terms of cells is bounded from above by $\frac{T^l}{t_l}+1$ under the timer-based LATS, while under the LI timer-based method is bounded from above by $\frac{T}{t_l}+1$. Hence, the maximal number of cells that need to be searched, under the timer-based LATS, in comparison to the LI timer-based method, is $O(\frac{T^l}{T})$ for very slow users (or equivalently for 1-D motion). For 2-D motion in Manhattan-like network the number of cells in a radius d is $O(d^2)$ for large d. Therefore, using an upper bound analysis, the superiority of the timer-based LATS over the LI timer-based method is $O([\frac{T^l}{T}]^2)$ for 2-D motion, or equivalently for fast moving users.

Below we show that, under certain conditions, these results are valid also for the *expected* cost of search. The expected

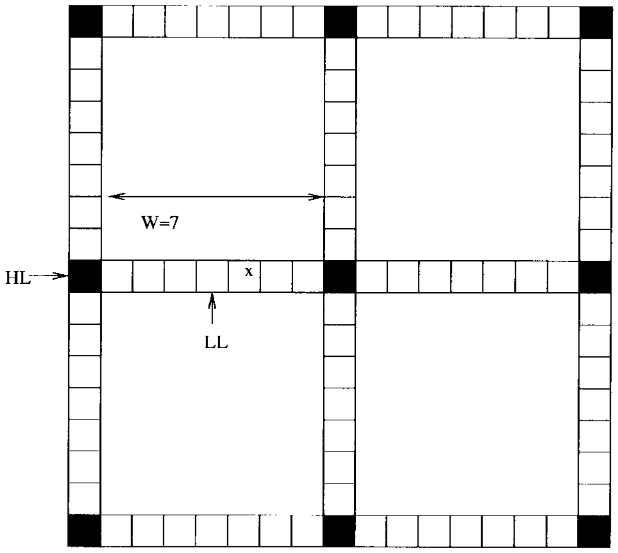


Fig. 2. System model.

number of cells that need to be searched depends, in general, on the user last known location, and on the length of the *roaming interval* τ. Due to the dependency of the registration process under LATS on the user location, the probability that the user last known location is the cell x, is not necessarily the steady state probability to find the user at x. Moreover, the distribution function of τ is not necessarily uniform. However, since call arrivals are generated as a Poisson process, independent of the user motion, for two extreme situations we can assume that τ is uniformly distributed over $[0, T^l-1]$. The first situation is for very slow user, such that $T^l < T < \min\{Wt_l, t_h\}$. The expected cost of search under the timer-based LATS is bounded from above by

$$E[\text{Search}_{\text{LATS}}] \leq \frac{1}{T^l}\sum_{\tau=0}^{T^l-1}\left(\frac{2\tau}{t_l}+1\right) = \frac{T^l-1}{t^l}+1. \tag{21}$$

Under the LI timer-based method, the expected cost of paging is given by

$$E[\text{Search}_{\text{LI}}] \approx \frac{1}{T}\sum_{\tau=0}^{T-1}\left(\frac{2\tau}{t_l}+1\right) = \frac{T-1}{t^l}+1. \tag{22}$$

Hence, we get that

$$\frac{E[\text{Search}_{\text{LATS}}]}{E[\text{Search}_{\text{LI}}]} \approx \frac{T^l}{T}. \tag{23}$$

The relation obtained in (23) can be easily generalized to a 1-D motion model. Note that the relation $\frac{t_h}{t_l}$ reflects the *load ratio* between HL cells and LL cells. In reality, the value of T^l is expected to be set to $T^l = \frac{Tt_l}{t_h}$. Hence, for most practical cases, it is expected that $\frac{T^l}{T} = \frac{t_l}{t_h}$. Substituting the last equality in (23) yields

$$\frac{E[\text{Search}_{\text{LATS}}]}{E[\text{Search}_{\text{LI}}]} \approx \frac{t_l}{t_h}. \tag{24}$$

Hence, for lowly mobile users, the expected reduction in paging cost is linearly proportional to the *load ratio*.

The other case in which we can assume a uniform distribution of τ is when the users move relatively fast. Assuming that $T^l \gg Wt_l + t_h$, the distance traveled since the last location update is, for a good approximation, the number of *blocks* traveled during τ time units, multiplied by $W+1$ (the number of cells in a block). Hence, the distance traveled is given by $d(\tau) \approx \lfloor \frac{\tau}{Wt_l+t_h} \rfloor (W+1)$. The PLA of the user in a distance d from its last known location x $(d \gg W+1)$ is a rhombus centered at x, with side length $\sqrt{2}d$. The number of cells need to be searched is, for a good approximation, equal to $2d^2$. Hence, for $\tau \gg Wt_l + t_h$ we get that

$$|\text{PLA}(\tau)| \approx 2\left[\left\lfloor \frac{\tau}{Wt_l + t_h} \right\rfloor (W+1)\right]^2. \tag{25}$$

The expected paging cost under the timer-based LATS is given by

$$E[\text{Search}_{\text{LATS}}] \approx \frac{1}{T^l} \sum_{\tau=0}^{T^l-1} |\text{PLA}(\tau)|. \tag{26}$$

Neglecting the first terms in the series, we use (25) to get

$$E[\text{Search}_{\text{LATS}}] \approx \frac{2}{T^l}\left[\frac{W+1}{Wt_l+t_h}\right]^2 \sum_{\tau=0}^{T^l-1} \tau^2 \tag{27}$$

which yields

$$E[\text{Search}_{\text{LATS}}] \approx \left[\frac{W+1}{Wt_l+t_h}\right]^2 \frac{(T^l-1)(2T^l-1)}{3} = O({T^l}^2). \tag{28}$$

Similarly, using that $T > T^l \gg Wt_l + t_h$, we get that for the LI timer-based method

$$E[\text{Search}_{\text{LI}}] \approx \left[\frac{W+1}{Wt_l+t_h}\right]^2 \frac{(T-1)(2T-1)}{3} = O(T^2). \tag{29}$$

Hence

$$\frac{E[\text{Search}_{\text{LATS}}]}{E[\text{Search}_{\text{LI}}]} \approx \left[\frac{T^l}{T}\right]^2. \tag{30}$$

Substitute $\frac{T^l}{T} = \frac{t_l}{t_h}$ in (30) yields

$$\frac{E[\text{Search}_{\text{LATS}}]}{E[\text{Search}_{\text{LI}}]} \approx \left[\frac{t_l}{t_h}\right]^2. \tag{31}$$

Equation (31) implies that for highly mobile users, the expected reduction in paging cost is quadratically proportional to the *load ratio*. For example, if the crossing time of an LL cell t_l is half of the crossing time of an HL cell t_h, then the timer-based LATS has the potential of reducing the expected paging cost by a factor of four, in comparison to the static timer method.

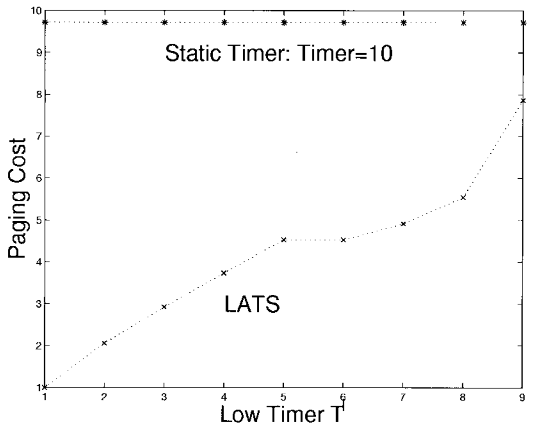


Fig. 3. Expected paging cost as a function of the low timer T^l $(t_l = 1)$.

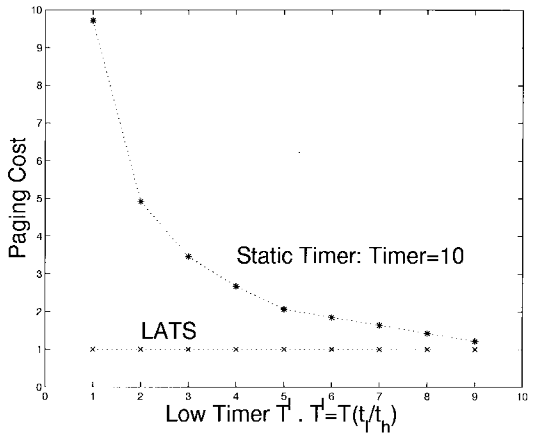


Fig. 4. Expected paging cost as a function of the *load ratio* t_h/t_l.

C. Comparison and Numerical Results

To demonstrate the performance of the LATS strategy, we examine several numerical examples, using a Manhattan-like system. Fig. 3 depicts the expected paging cost under the timer-based LATS and the static timer-based method, as a function of the low timer T^l, where $t_l = 1$, $t_h = 10$, $W = 7$, and $T = 10$. Clearly, the LATS strategy outperforms the static timer method. The performance ratio is, for a good approximation, proportional to the ratio $\frac{T^l}{T}$.

Fig. 4 depicts the expected paging cost as a function of the *load ratio* $\frac{t_h}{t_l}$. The value of t_l ranges from 1 to 9, thus affecting the value of T^l, which is set to $T^l = \frac{Tt_l}{t_h}$, $t_h = 10$. The expected paging cost under both strategies is plotted as a function of t_l. Note that the ratio $\frac{T^l}{t_l}$ remains constant, since $\frac{T^l}{t_l} = \frac{T}{t_h} = 1$. Hence, the performance of the LATS strategy is insensitive to t_l, while the performance of the static timer method improved with t_l. When t_l approaches t_h, the performance of the static timer method approaches to the performance of the LATS strategy, as expected.

Fig. 5 depicts the performance ratio $\frac{E[\text{Search}_{\text{LATS}}]}{E[\text{Search}_{\text{LI}}]}$, as a function of the *load ratio* $\frac{t_h}{t_l}$ for a lowly mobile user (i.e. a very slow user) and for a highly mobile user. The parameters t_l, t_h are the same as used in Fig. 4. For the lowly mobile user $T = 10$, while for the highly mobile user $T = 500$. For

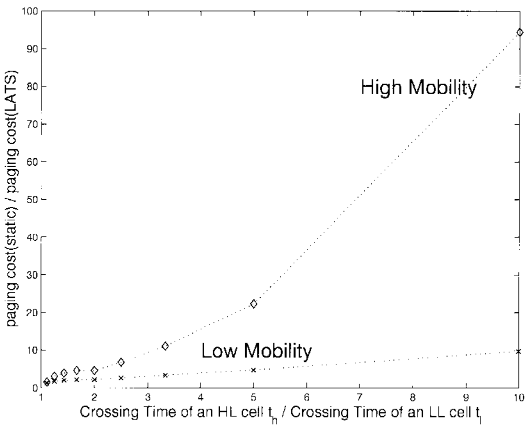


Fig. 5. The performance ratio $\frac{\text{paging cost(static)}}{\text{paging cost(LATS)}}$ as a function of the *load ratio* t_h/t_l.

both users $T^l = T\frac{t_l}{t_h}$. The performance ratio is linear with the load ratio for a lowly mobile user and quadratic for a highly mobile user.

V. Summary and Concluding Remarks

An LATS for tracking mobile users in wireless networks was introduced. It provides a simple mechanism which enable the system to dynamically adjust the registration activity to the system load, in order to reduce the wireless cost of paging. The tracking strategy is based on a local *registration threshold* (e.g., timer, distance$\cdots$), which varies from one cell to another, as a function of the local load on the cell. Our analysis shows that the LATS strategy can significantly reduce the paging cost, in comparison with the LI (static) timer-based method, without increasing the *actual* wireless cost of registration. This is done by using nonutilized wireless resources for increasing the registration activity at idle epochs. The main advantage of the LATS strategy over a static method is its ability to use non utilized system resources. In reality, a static timer is adapted to the maximal local load in the network, at the busy hour. Thus, for most of the time it is sub-optimal for most of the cells in the network. The LATS strategy suggests a significant improvement not only at LL cells, as someone would intuitively expect, but also at highly loaded cells. This is done by eliminating unnecessary searches after users residing in LL cells, and by forcing the user to register while crossing an LL zone. Moreover, if higher user density is coupled with less mobility (as is the case for vehicular users), then the LATS strategy offers further performance improvement.

Appendix A
Proof of Lemma 3.1

Consider a 2-D grid system, where the local timer at each cell changes randomly every time slot, such that the local timer can take any integer value in the interval $[1, \tau]$, with equal probability for each value. At every time slot, the user can move, with equal probability, to each one of its eight nearest neighbors. Thus, the number of cells reachable within τ time slots is $O(\tau^2)$, meaning that after τ steps, there are $O(\tau^2)$ candidates for the user location at time τ. A cell x is a member in the PLA if and only if there exists at least one feasible roaming interval in length τ, describing the user locations at time $t = 0, 1, 2, \cdots \tau$. Thus, given a cell x, the verification that $x \in \text{PLA}$ has a computational complexity of $O(\tau)$, since exactly τ vertices in the mobility graph must be verified. Since the size of the PLA is $O(\tau^2)$, its verification required at least $O(\tau^3)$ steps. Thus, constructing the PLA must have a computational complexity of $\Omega(\tau^3)$. □

Appendix B

The goal is to show that the following expression is bounded from above by $\frac{T+1}{2}$:

$$S = \sum_{i=0}^{T-1} \pi_i \frac{1}{T_i} \sum_{\tau=0}^{T_i-1} \min\{(2\tau+1), 2(T-\tau)\}$$
$$= \frac{2}{T(T+1)} \sum_{i=0}^{T-1} \sum_{\tau=0}^{T-i-1} \min\{(2\tau+1), 2(T-\tau)\}. \quad (32)$$

Let us denote $\theta(T_i) = \sum_{\tau=0}^{T_i-1} \min\{(2\tau+1), 2(T-\tau)\}$. For $T_i \le \lceil \frac{T}{2} \rceil$, we get that

$$\theta(T_i) = \sum_{\tau=0}^{T_i-1} (2\tau+1) = T_i^2 \quad \forall T_i \le \left\lceil \frac{T}{2} \right\rceil. \quad (33)$$

For $T_i = T$, we get that

$$\theta(T) = \sum_{\tau=0}^{\lceil \frac{T}{2} \rceil - 1} (2\tau+1) + \sum_{\tau=\lceil \frac{T}{2} \rceil}^{T-1} 2(T-\tau)$$
$$= \sum_{\tau=0}^{\lceil \frac{T}{2} \rceil - 1} (2\tau+1) + \sum_{i=1}^{\lfloor \frac{T}{2} \rfloor} 2i = \frac{T(T+1)}{2}. \quad (34)$$

For $T_i > \lceil \frac{T}{2} \rceil$, we get that

$$\theta(T_i) = \theta(T) - \sum_{\tau'=T_i}^{T-1} 2(T-\tau')$$
$$= \theta(T) - (T-T_i)^2 - (T-T_i). \quad (35)$$

Substitute (33)–(35) in (32), we get that

$$S = \frac{2\sigma}{T(T+1)} \quad (36)$$

where $\sigma = \sum_{T_i=1}^{\lceil \frac{T}{2} \rceil} T_i^2 + \sum_{T_i=\lceil \frac{T}{2} \rceil+1}^{T} [\theta(T) - (T-T_i)^2 - (T-T_i)]$. Substitute $\Delta = T - T_i$, and $\theta(T) = \frac{T(T+1)}{2}$ in (36), we get that

$$S = \frac{2}{T(T+1)} \left[\sum_{T_i=1}^{\lceil \frac{T}{2} \rceil} T_i^2 + \frac{T(T+1)(\lfloor \frac{T}{2} \rfloor)}{2} - \sum_{\Delta=0}^{\lfloor \frac{T}{2} \rfloor - 1} (\Delta^2 + \Delta) \right]. \quad (37)$$

Subtracting the summation on Δ^2 from the summation on T_i^2 we get that for T odd

$$S_{\text{odd}} = \left\lfloor \frac{T}{2} \right\rfloor + \frac{2(\lceil \frac{T}{2} \rceil^2 + \lfloor \frac{T}{2} \rfloor^2) - \lfloor \frac{T}{2} \rfloor (\lfloor \frac{T}{2} \rfloor - 1)}{T(T+1)}. \quad (38)$$

Using that for T odd, $\lfloor \frac{T}{2} \rfloor = \frac{T-1}{2}$ and $\lceil \frac{T}{2} \rceil = \frac{T+1}{2}$, and that $T > 1$, we get that

$$S_{\text{odd}} = \frac{T-1}{2} + \frac{4(T^2+1) - (T-1)(T-3)}{4T(T+1)}. \tag{39}$$

Hence

$$S_{\text{odd}} = \frac{T-1}{2} + \frac{3T^2+4T+1}{4T^2+4T} < \frac{T-1}{2} + 1 = \frac{T+1}{2}. \tag{40}$$

Similarly, using that for T even $\lfloor \frac{T}{2} \rfloor = \lceil \frac{T}{2} \rceil = \frac{T}{2}$, (37) implies that for T even

$$\begin{aligned} S_{\text{even}} &= \frac{T}{2} + \frac{T^2}{2T(T+1)} - \frac{T-2}{4(T+1)} \\ &= \frac{T}{2} + \frac{T+2}{4(T+1)} < \frac{T}{2} + \frac{T+2}{2T+4} = \frac{T}{2} + \frac{1}{2} = \frac{T+1}{2}. \end{aligned} \tag{41}$$

Equations (40) and (41) imply that $S < \frac{T+1}{2}$. □

References

[1] A. Bar-Noy and I. Kessler, "Tracking mobile users in wireless networks," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1877–1886, Nov. 1993.

[2] A. Bar-Noy, I. Kessler, and M. Sidi, "Mobile users: To update or not to update?," *Wireless Networks*, vol. 1, no. 2, pp. 175–185, 1995.

[3] J. Ho and I. F. Akyildiz, "Mobile user location update and paging under delay constraints," *Wireless Networks*, vol. 1, pp. 413–425, 1995.

[4] L. Kleinrock, *Queueing Systems*. New York: Wiley, 1976, vols. 1, 2.

[5] H. Levy and Z. Naor, "Active tracking: Locating mobile users in personal communication service networks," *ACM J. Wireless Networks*, to be published.

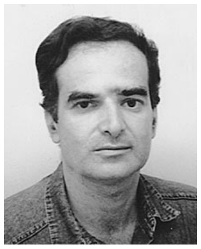
[6] U. Madhow, L. Honig, and K. Steiglitz, "Optimization of wireless resources for personal communications mobility tracking," *IEEE Trans. Networking*, vol. 3, pp. 698–707, Dec. 1995.

[7] C. Rose and R. Yates, "Minimizing the average cost of paging under delay constraints," *ACM J. Wireless Networks*, vol. 1, pp. 211–219, 1995.

[8] C. Rose, "Minimizing the average cost of paging and registration: A timer-based method," *ACM J. Wireless Networks*, vol. 2, no. 2, pp. 109–116, 1996.

[9] S. Tabbane, "An alternative strategy for location tracking," *IEEE J Select. Areas Commun.*, vol. 13, pp. 880–892, May 1995.

[10] EIA/TIA, "Cellular radio-telecommunications inter-system operations," Tech. Rep. IS-41 Revision C, 1995.



Zohar Naor (S'98) received the B.Sc. and M.Sc. degrees in computer science from Tel-Aviv University, Israel, in 1987 and 1992, respectively, where he is currently working toward the Ph.D. degree in computer science.

He has worked in the areas of digital communication and wireless networks. His current research interests include wireless networks, computer communication networks, and multiple access.

Hanoch Levy (S'83–M'87), for photograph and biography, see this issue, p. 936.