Location Management in Mobile Computing

Evaggelia Pitoura

Computer Science Department,
University of Ioannina,
Ioannina, Greece
http://www.cs.uoi.gr/~pitoura

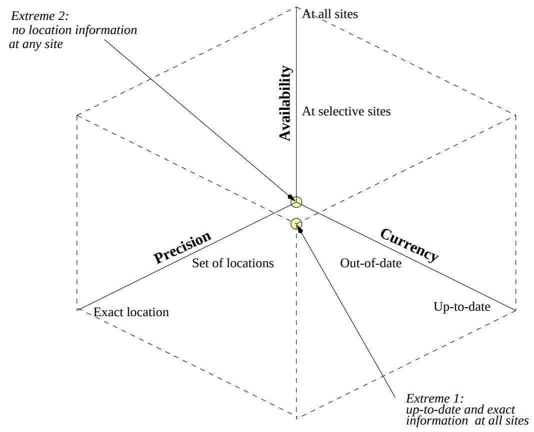
Summer School, Jyvaskyla, August 1998

Introduction

Applications

- tied to wireless hardware (mobile users)
- mobile software, i.e., code or data (migration, mobile agents, ubiquitous computing)

Taxonomy



Infrastructure

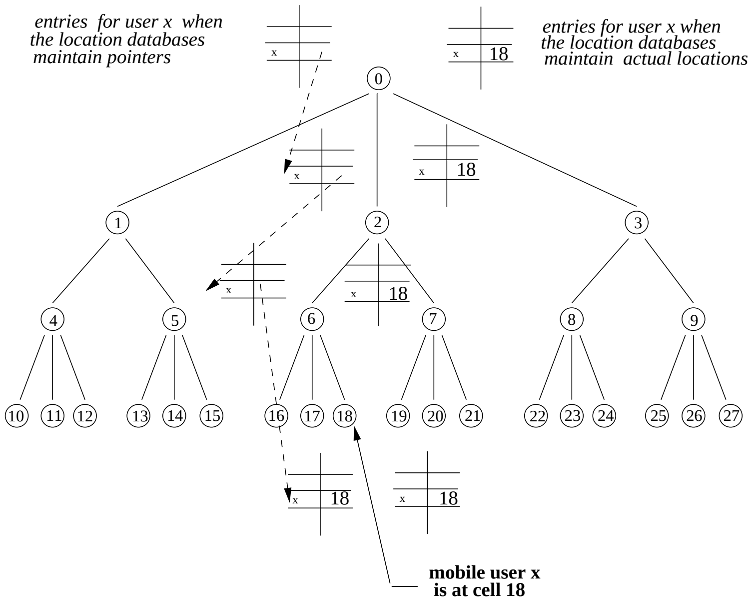
- Cellular Architecture – WAN – LAN
- GPS

Architectures

Two-Tier

Home Location Register (HLR)
Visitor Location Registers (VLR)

Hierarchical Schemes



Comparison

(+)	No need for life-long numbering (no pre-assigned HLR)
(+)	Support for locality
(-)	Increased number of operations (database operations and communication messages)
(-)	Increased load and storage requirements at higher-levels

Placement of Databases

Entries at the Leaves (VLRs)

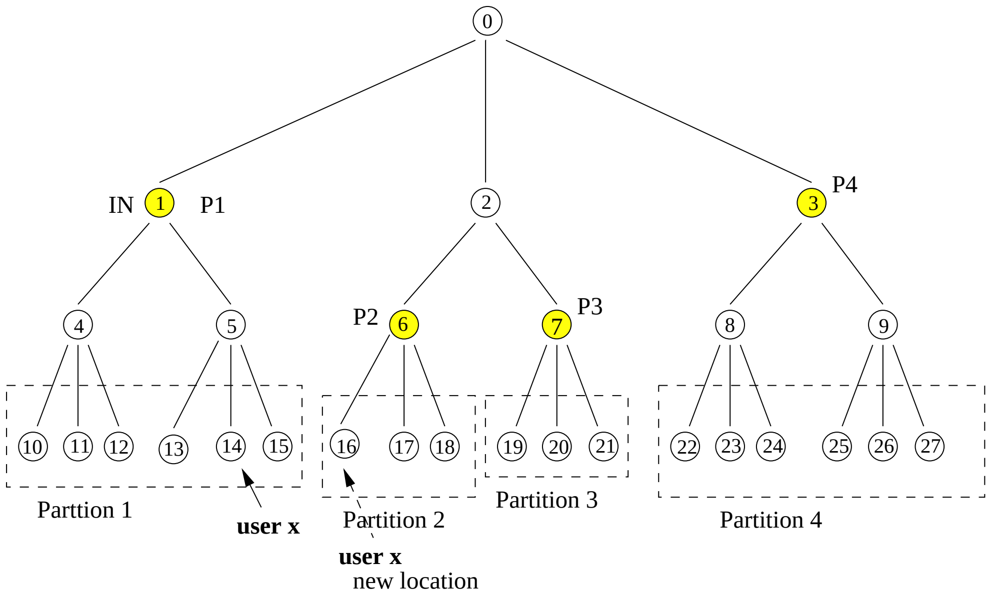
flat, expanding, hybrid

Optimization

Objective functions: (a) the number of database updates and accesses, (b) the communication cost, (c) the sum of the traffic on the network link or links.

Constraints (a) database capacity (b) link capacity, and (c) storage.

Partitions



Caching

Two-Tier

After a call, save location at the caller's VLR

Invalidation
eager caching
lazy caching

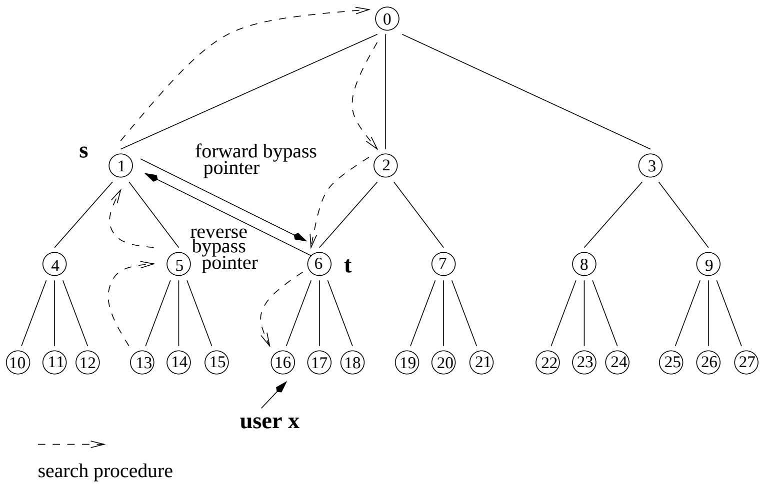
Performance
a hit ratio threshold $p_T = C_H / C_B$, where C_H is the cost of a lookup when there is a hit and C_B the cost of the lookup in the non-caching scheme. Among other factors, C_H and C_B depend on the relative cost of querying HLR's and VLR's.

In practice, it is expected that $LCMR_T > 7$

Other
replacement? initialization?

Caching

Hierarchical



Variations

- simple caching – level caching
- lazy –
- exact locations – pointers

Performance

Regional Call-to-Mobility Ratio (RCMR) for users with $RCMR > 5$, a 30% reduction when considering only the number of database operations.

More on granularity: caching and partitions

Replication

Replicate the location of specific users at selected sites

judicious replication of i at j

$$\alpha * C_{i,j} \geq \beta * U_i \tag{1}$$

α: cost savings when a local lookup, as opposed to a remote query, succeeds
β: replica update cost
$C_{i,j}$: expected number of calls from j to i over time T, and
U_i: number of moves made by i over T.

Other Factors:
database service capacity, storage

Other Issues:

- where to keep replication set
- other applications
- *granularity* of location replicas

Comparison with file allocation [3]and database allocation [12] problem.

Replication

1. Per User Profile Replication [17]

Problem Formulation

Let M: the number of users and N: number of zones. Find a replication assignment of a user's profile P_i to a set of zones $R(P_i)$ such that the system cost is minimized:

$\Sigma_{i=1}^{N} \Sigma_{j=1,Z_j \in R(P_i)}^{M} \beta * U_i - \alpha * C_{i,j}$

given constraints on the maximum number p_j of replicas per zone Z_j and on the maximum number of replicas r_i per user P_i.

Solution: Construct a flow network F

Vertices: source vertex s, sink vertex t, users P_i and zones Z_j

Edges:

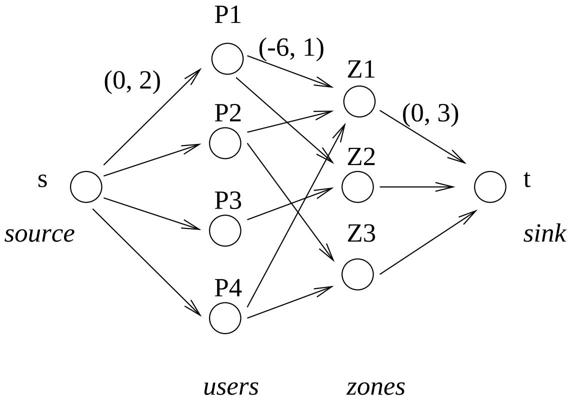
A pair (c, p) of attributes with each edge

$s \rightarrow P_i$, with $(c, p) = (0, r_i)$

$Z_j \rightarrow t$ with $(0, p_j)$

$P_i \rightarrow Z_j$ with $(c, p) = (\beta * U_i - \alpha * C_{i,j}, 1)$ iff it is judicious to replicate P_i at Z_j,

Compute a minimum-cost maximum-flow on F



Replication

Extensions
Adaptation to changing calling and mobility patterns
Compute $F_{\tau_{new}}$ from $F_{\tau_{old}}$.

2. Working Set Replication [14]

Assumption: each user communicates frequently with a small number of sources, called its working set, $\Rightarrow$ maintain copies of its location at the members of this set.

Similar to the per-user replication but no constraints, thus the decision whether to replicate P_i at Z_j made independently at each unit P_i
Evaluate Inequality (**??**) locally at the mobile unit P_i when:

- a call is set up and the caller's site is not a member of the working set of the callee
  If (1) holds, the caller enters the set
- P_i moves.
  Re-evaluate (1) for all members of the working set
  Drop a member, if (1) does not hold

Replication

3. Replication in Hierarchical Architectures

Note: databases at higher levels tend to be selected as replication sites over databases at lower levels,

HiPer[7]

- It is shown that: it is never judicious to replicate i at j if $LCMR_{i,j} < R_{min}$, while it is always judicious to replicate, if $LCMR_{i,j} \geq R_{max}$.
- Constraints: N_{max}: maximum number N_{max} of replicas per user and a cap L on the maximum level at which locations may be replicated.
- off line algorithm that proceeds in two phases:
  1. in a bottom-up traversal, allocate replicas of i at all databases with $LCMR_{i,j} \geq R_{max}$ as long as the number of allocated replicas n does not exceed N_{max}.
  2. if $n \leq N_{max}$, in a top-down traversal, allocate the remaining replicas to databases below level L with the largest non negative

Replication

4. The Adaptive Data Replication (ADR) Algorithm [21]

Presents a solution to the general problem of determining an optimal (in terms of communication cost) set of replication sites for an object in a distributed system, when the object's read-write pattern changes dynamically.

Preliminaries

- tree-structure architectures
- R: the current replication set of object x
- A site i is an $\overline{R}$-*neighbor*, if it belongs to R but has a neighbor site that does not belong to R.
- When site R is not a singleton set, a site i is an R-$fringe$ site, if it is a leaf at a subgraph induced by R.

The Algorithm

- R is updated periodically every T, specifically every T three tests are performed:
- The *expansion test* performed by each $\overline{R}$-*neighbor* site i Site i invites each of its neighbor j not in R to join R, if the number of reads that i received from j during the last T is greater the number of writes that i received during T from i itself or from a neighbor other than j.

Replication

The ADR Algorithm (continue)

- The *contraction* test executed by each *R-$fringe$* site i
  Site i requests permission from its neighbor site j in R to exit R, if the number of writes that i received from j during T period is greater than the number of reads that i received during T.
- If site i is both an $\overline{R}$-*neighbor* and an $R - fridge$, it executes the expansion test first, and if the test fails (i.e., no site joins R), then it executes the contraction test.
- The *switch* test is executed, when R is a singleton test and the expansion test that the single site i in R has executed fails.
  Site i asks a neighbor site n to be the new singleton site, if the number of requests received by i from n during T is larger than the number of all other requests received by i during T

The ADR algorithm is shown to be *convergent-optimal*: starting at any replication scheme, it converges to the replication scheme that is optimal to the current read-write pattern.
The convergence occurs within a number of time periods that is bounded by the diameter of the network.

Forwarding Pointers

When the number of moves that a user makes is large relative to the number of calls it receives, defer updating database entries holding the user's location

Two-tier Architectures [5]

x's HLR is not updated, each time x moves to a new location.

leave a forwarding pointer at the VLR at x's previous location to point to the VLR at the new location.

Calls follow a chain of forwarding pointers.

The length of the chain of forwarding pointers grows up to a maximum value of K.

Since the approach is applied on a per-user basis, the increase in the cost of call operations affects only the specific user.

The router optimization extensions to IEFT Mobile IP protocol include pointer forwarding in conjunction with lazy caching [8].

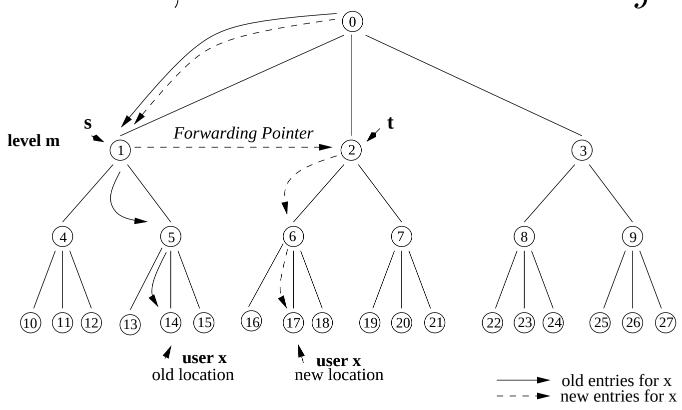
Performance depends on the cost of setting up and traversing pointers relative to the costs of updating the HLR. An analytical estimation [5]: under certain assumptions and if pointer chains are kept short ($K < 5$), forwarding can reduce the total network cost by 20%-60% for users with $CMR < 0.5$

Forwarding Pointers

Hierarchical Architectures

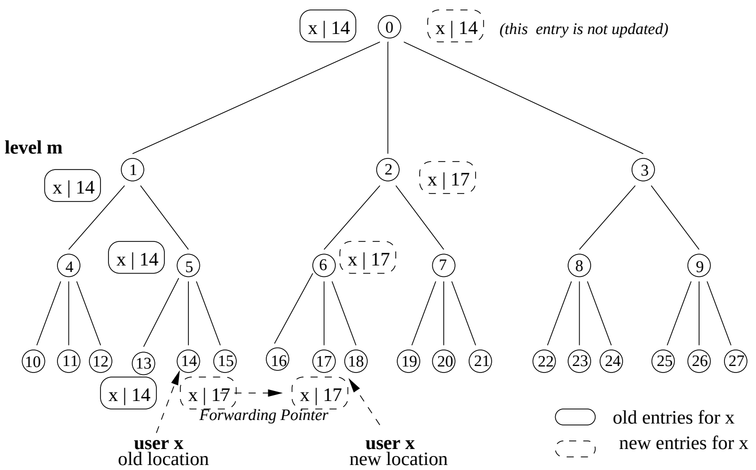
When x moves from i to j, instead of updating all databases on the path from j through $LCA(j, i)$ to i, only the databases up to a level m are updated.

A forwarding pointer is set from node s to node t, where s is the ancestor of i at level m, and t is the ancestor of j at level m.



simple forwarding vs level forwarding

When entries at the internal nodes are actual addresses



Forwarding Pointers

An analysis of a forwarding method when entries are actual addresses [10] along with caching based on he degree of mobility (CMR) host (low or high) and on whether it has a large number of frequent callers

Updating obsolete entries in databases at levels higher than m: e.g., after a successful lookup, or each node sends a location update message to all location servers on the path to the root during off-peak hours.

Pointer reduction [13]

Applications in Software Systems to maintain references to mobile objects:

Emerald [9] is an object-based system in which objects move within the system.

SSP chains [16] are chains of forwarding pointers for transparently migrating object references between processes. SSP uses a short-cutting technique.

Taxonomy

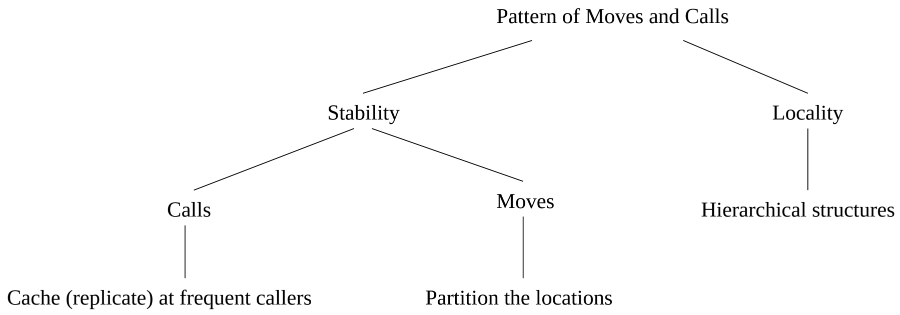
Exploit knowledge about the calling and moving behavior of mobile objects: *stability* and *locality*.

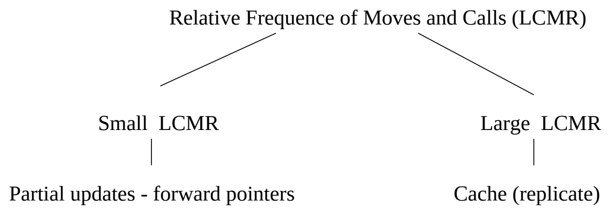
Stability of calls: most calls for a user originate from the same set of locations.

Stability of moves: users tend to move inside specific regions.

Locality: the cost of a lookup or update operation increases with the distance. Local operations (moves to neighbor locations or calls from near-by places) are common and should cost less than remote operations.

Relative frequency of calls and moves, since often decrease the cost of either the move or call operation in the expense of the other.





Taxonomy

More specific types of movement and calling: e.g., follow a certain mobility pattern or there is an epicenter (e.g., home location) of movement.

Models of movement can be used in guiding the search for the current location of a mobile object (see for example, [15, 1]). For instance, search candidate location in descending order of the probability of the user being there.

Dynamic adaptation to the current pattern and ratio.

Employment on a per user basis – overall – per group of users (e.g., based on their geographical location or on their mobility and calling characteristics) all users that receive a large number of calls) or a combination of both.

Variations
- Dynamic (adaptive) or static
- Per object, group of objects, geographical region

The topology of network sites, how they are populated and their geographical connectivity.

Scales with the number of mobile objects, operations and geographical distribution.

Taxonomy

Estimation of the current value of the CMR

- The *running average algorithm* [6]:
  maintain for every user the running counts of the number of incoming calls and the number of times that the user changes location.
- store information about the CMR, for instance in the HLR, and download it during off-peak hours.
- analytical estimations
  For instance, if the coming call stream to a user is consider a Poisson process with arrival rate λ and the time a user resides in a region has a general distribution with mean $1/\mu$, then $LCMR = \lambda/\mu$.
- Traces of actual moving users (for example, (SUMATRA)[19].

Evaluation based on database operations:
minimizing (a) the total number of database updates and queries, (b) the database load and size, and (c) the latency of each database operation.

and communication:
reduce among others (a) the total number of messages, (b) the number of hops, (c) the distance traveled, (d) the number of bytes generated, and (e) the sum of the traffic on each link or over all links.

Taxonomy

Two-Tier Schemes

Method	Variations		Applicable when:
Caching When x is called by y, cache x's location at y's zone	*Eager caching:*	Cache update overhead occurs at moves	Large LCMR Call Stability
	Lazy caching:	Cache update overhead occurs at calls	
Replication Selectively replicate x's address at the zones from which it receives the most calls	*Per-user Profile Replication:*	Additional constraints are set on the number of replicas per site and on the number of replicas per user	Large LCMR Call Stability
	Working Set:	Adaptive and distributed: the replication sites are computed dynamically by each mobile host locally	
Forwarding Pointers When x moves, add a forwarding pointer from its old to its new address	Restrict the length of the chain of forwarding pointers		Small LCMR

Hierarchical Schemes

Method	Issues/Variations	Appropriate when:
Caching When x at zone i is called by user y at zone j, cache at a node on the path from j to LCA(i, j) a pointer to a node on the path from i to LCA(i, j) to be used by any subsequent call to x from zone j.	Up to which tree level to maintain cache entries When to update cache entries	Large CMR Call Stability
Replication Selectively replicate x's location at internal and/or leaf databases.		Large CMR Call Stability
Forwarding Pointers When x moves from cell i to cell j, instead of updating all databases on the path from i to LCA(i, j) and from LCA(i, j) to j, update all databases up to some level m and add a forwarding pointer at the level m ancestor of i to point to the level m ancestor of j.	When and how to purge the forwarding pointers Setting the level m	Small LCMR
Partitions Divide the locations into sets (partitions) so that the user moves inside a partition frequently and crosses the boundary of a partition rarely. Keep information about the partition in which the user resides instead of its exact location		Move Stability

Concurrency Control

Moves and calls are issued asynchronously and concurrently and each results in number of database operations ⇒ concurrency control to ensure correctness

leave a forwarding pointer to the new location

- when a call reads obsolete data and fails, it is reissued. No upper bound on the number of attempts.
- traditional database concurrency control techniques such as locking or timestamps
- impose a specific order on the execution of the operations
  - first, add entries at the path from j to $LCA(i, j)$ in a bottom-up fashion
  - then, delete the entries at the path from the $LCA(i, j)$ to i in a top-down fashion.
  - Special care so that during the delete phase, an entry at a level $k - 1$ is deleted only after servicing all lookups from higher-level databases.
  - [2]: application to the regional matching method

When **replication** ⇒ coherency control protocols to maintain the replicas consistent

– an HLR or a master copy that is always consistent

– use forwarding pointers to handle any incoming calls directed there from obsolete replicas.

Failure Recovery

Database recovery after the failure of a location database.

VLR Failure Restoration

Periodic Checkpointing

- If the VLR is checkpointed, the backup record is recovered.
- But if the backup is obsolete, then all areas within the VLR must be paged to identify the mobile users currently in the VLR's zone. Thus no improvement.
- GSM exercises periodic location updating: the mobile users periodically establish contact with the network to confirm their location.
- Periodic confirmation does not improve the restoration process, if the confirmation frequency < 0.1 times of the portable moving rate [11].

Location Update on Demand [11]

- Eliminates the need for periodic confirmation messages.
- After a failure, a VLR restoration message is broadcasted to all mobile users in the area associated with the VLR.
- The mobile users then send a confirmation message. To avoid congesting the base station, each such message is sent within a random period from the receipt of the request.

Failure Recovery

HLR Failure Restoration

In GSM,

- the HLR database is periodically checkpointed. After an HLR failure, the database is restored by reloading the backup.
- If the backup is obsolete, calls are lost.
- Obsolete data are updated by either a call origination or a location confirmation

In IS-41,

- After an HLR failure, the HLR sends an "Unreliable Roamer Data Directive" to all associated VLRs.
- The VLRs remove all records of associated with that HLR.
- Later, when a portable is registered at a VLR, the VLR sends a registration message to the HLR allowing it to be incrementally reconstructed. Before, calls are lost.

Aggressive Restoration [11]

- HLR restores its data by requesting all the VLRs referenced in its backup copy to provide exact location information
- An algorithm to identify VLRs that are not mentioned in the backup; e.g., VLRs such that there are portables that moved in between the last HLR checkpointing and the failure and not out

Location Queries

advanced queries that involve the location of moving objects

Examples: finding the nearest service, or identifying the shortest route with the best traffic conditions.

- may not include location directly, but may require tracking mobile objects *indirectly*, e.g., queries that involve data produced and located at mobile hosts.
- may be imposed by either static or mobile users and may include databases located at both static and mobile sites.
- have both a *spatial* dimension, e.g., involve the position of a user and a *temporal* dimension, e.g., involve time,
- may include *transient data*, that is data whose value changes while the queries are being processed, e.g., a moving user asking for nearby hospitals.
- *continuous queries*, e.g., a moving car asking for hotels locating within a radius of 5 miles and requesting the answer to the query to be continuously updated. Issues related to continuous queries include when and how often should they be re-evaluated and the possibility of a partial or incremental evaluation.
- *Imprecision*

Location Queries

Bounded Ignorance

How to derive an optimal execution plan for a location query that will acquire only the missing information necessary to answer it [4].

Partitions

The system guarantees *bounded ignorance*: in that the actual and stored location of a user are always in the same partition.

To determine the actual location of a user, searching in the partition of its stored location is sufficient.

Deriving an optimal execution plan reduces to determining an optimal sequence in which to search inside the partitions of the users involved in the query.

Location Queries

Continuous Queries

The position of a moving object is represented as a function of time. [18].

Thus, position changes *continuously* with time even without an explicit update through a database operation.

A new data model, called MOST, is proposed to incorporate such *dynamic attributes*.

MOST enables queries that refer to future values of dynamic attributes, e.g., retrieve all the airplanes that will come within 30 miles in the next 10 minutes.

The answer to future queries is tentative.

Routes [20]

Objects move on predefined routes.

The current position of an object is modeled as the distance from its starting point along a given route.

Indexing the location of moving objects.

References

[1] I. F. Akyildiz and J. S. M. Ho. Dynamic Mobile User Location Update for Wireless PCS Networks. *ACM/Baltzer Wireless Networks Journal*, 1(2), 1995.

[2] B. Awerbuch and D. Peleg. Concurrent Online Tracking of Mobile Users. In *Proceedings of SIGCOMM 91*, pages 221–233, November 1991.

[3] L. W. Dowdy and D. V. Foster. Comparative Models of the File Assignment Problem. *ACM Computing Surveys*, 14(2):288–313, June 1982.

[4] T. Imielinski and B. R. Badrinath. Querying in Highly Mobile Distributed Environmnets. In *Proceedings of the 18th International Conference on Very Large Data Bases (VLDB 92)*, 1992.

[5] R. Jain and Y-B. Ling. A Auxiliary User Location Strategy Employing Forwarding Pointers to Reduce Network Impacts of PCS. *Wireless Networks*, 1:197–210, 1995.

[6] R. Jain, Y-B. Ling, C. Lo, and S. Mohan. A Caching Startegy to Reduce Network Impacts of PCS. *IEEE Journal on Selected Areas in Communications*, 12(8):1434–44, October 1994.

[7] J. Jannink, D. Lam, N. Shivakumar, J. Widom, and D.C. Cox. Efficient and Flexible Location Management Techniques for Wireless Communication Systems. *ACM/Baltzer Journal of Mobile Networks and Applications*, 3(5):361–374, 1997.

[8] D. B. Johnson and D. A. Maltz. Protocols for Adaptive Wireless and Mobile Networking. *IEEE Personal Communications*, 3(1), 1996.

[9] E. Jul, H. Levy, N. Hutchinson, and A. Black. Fine-Grained Mobility in the Emerald System. *ACM Transactions on Computer Systems*, 8(1):109–133, February 1988.

[10] P. Krishna, N. H. Vaidya, and D. K. Pradhan. Static and Dynamic Location Management in Mobile Wireless Networks. *Journal of Computer Communications (special issue on Mobile Computing)*, 19(4), March 1996.

[11] Y-B. Ling. Failure Restoration of Mobility Databases for Personal Communication Networks. *Wireless Networks*, 1:367–372, 1995.

[12] M. T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice Hall, 1991.

[13] E. Pitoura and I. Fudos. An Efficient Hierarchical Scheme for Locating Highly Mobile Users. In *Proceedings of the 7th Iternational Conference on Information and Knoweledge Management (CIKM'98)*, November 1998. To appear.

[14] S. Rajagopalan and B. R. Badrinath. An Adaptive Location Management Strategy for Mobile IP. In *Proceedings of the 1st ACM International Conference on Mobile Cmputing and Networking (Mobicom'95)*, Berkeley, CA, October 1995.

[15] C. Rose and R. Yates. Location Uncertainty in Mobile Networks: a Theoretical Framework. *IEEE Communications Magazine*, 35(2), 1997.

[16] M. Shapiro, P. Dickman, and D. Plainfosse. SSP Chains: Robust, Distributed References Supporting Acyclic Garbage Collection. Technical Report Technical Report 1799, INRIA, Rocquentcourt, France, November 1992.

[17] N. Shivakumar and J. Widom. User Profile Replication for Faster Location Lookup in Mobile Environments. In *Proceedings of the 1st ACM International Conference on Mobile Cmputing and Networking (Mobicom'95)*, 161-169, October 1995.

[18] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. In *Proceedings of the 13th International Conference on Data Engineering (ICDE 97)*, 1997.

[19] Stanford Pleiades Research Group. Stanford University Mobile Activity TRAces (SUMATRA). www-db.stanford.edu/sumatra.

[20] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and Imprecision in Modeling the Position of Moving Objects. In *Proceedings of the 14th International Conference on Data Engineering (ICDE 98)*, 1998.

[21] O. Wolfson, S. Jajodia, and Y. Huang. An Adaptive Data Replication Algorithm. *ACM Transactions on Database Systems*, 22(2):255–314, June 1997.