Concurrent Online Tracking of Mobile Users

Claudio Munari

Seminar of Distributed Computing
WS 03/04

The Paper

- Concurrent online tracking of mobile users
  B. Awerbuch, D. Peleg
  SIGCOMM 1991

The „full paper":

- Online Tracking of Mobile Users
  B. Awerbuch, D. Peleg
  Journal of the ACM, 1995

Outline

1. The Location Problem

1. The Location Problem

- Key problem of a communication network: **location of entities** in order to route traffic
- Handled by **name servers** / **distributed directories**
- How to locate mobile users in (fixed) wireless networks?
- How to deal with mobile entities that travel from one network site to another?
- Need of **dynamic mechanism** to keep track of such moving entities

Operations needed for a tracking mechanism

- „**move**" operation
  inform system about new address
- „**find**" operation
  locate user at his current address

How can the communication overhead of „move" and „find" operations be minimized?

Tracking Strategies

Conventional approaches:

- Single, central database
- **GSM**: Current location information of a user is stored in his home location register (HLR).
- **Mobile IP**: home agent keeps track of current care-of address

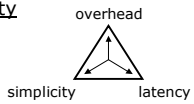
→ Not efficient in large, dynamic networks

7

Tracking Strategies (2)

Distributed approaches:

- **„Full-information" Strategy**
  Every node maintains a complete directory
- **„No-information" Strategy**
  No directories. Find users by global search
  Alternative: Forwarding pointers
- **„Partial-information" Strategy**
  → Trade-off between overhead, latency and simplicity



8

„Partial-information" Strategy

Goal:

- Strategy, that will perform well for any communication/travel pattern, making the costs of both „move" and „find" operations relatively cheap.

9

„Partial-information" Strategy

Ideas:

- Moves to near-by locations and searches for near-by users should be cheap
- Decomposition of network into regions
- **Hierarchy of regional directories**
  - hierarchies are the most general and scalable solution in large organizational problems
  - **locality propertiy**: exploit the fact that many moves are local

10

Hierarchy of regional directories

- Built of sets of network nodes: **Read** and **Write** sets
- Set construction based on graph-theoretic notion of a **regional matching**.

+ Low worst-case communication cost
+ Local operations always cost less than global ones.

The i'th level regional directory tracks any user residing within distance 2^i from a searching node

11

Why a limitation of 2^i?

Limiting the operational range of the i'th level of a regional directory to radius 2^i allows to:

- make the costs of finding a user proportional to 2^i.
- **bind the cost** of a „find" operation to a function of the user's distance from the searcher.
- **avoid constant updates** due to „move" operations of users far away.

12

Organization of a Regional Directory

Organization based on a **m-regional matching**:

- Collection of sets of nodes, consisting of a read set **Read(v)** and a write set **Write(v)** for every $v \in V$.
- The collection RW of all pairs of read and write sets,
  RW = {Read(v), Write(v) | $v \in V$},
  is an m-regional matching, if

$$\text{Write}(v) \cap \text{Read}(u) \neq \emptyset$$

for all $v, u \in V$, s.t. $dist(u,v) \leq m$.

2. Model and Main Results of the Paper

The Model

- Connected, undirected, weighted graph G = (V,E,w), |V| = n
- **Weight w(e)** represents the length / cost of transmitting a basic message (of length O(log n) on edge e
- Bidirectional communication channels
- Assume **efficient routing facilities** provided by the system
- **dist(u,w):**
  length (weight) of shortest path between u and w
- **diameter D(G):**
  maximal distance between any two vertices in G
- Operations are performed **concurrently** and **asynchronously**

The Model (2)

- **Addr(ξ)**: current address of user ξ
- **Directory Server D**: distributed data structure supporting the operations:
  - Find(ξ,v)
  - Move(ξ,s,t)

What is the „**amortised overhead**" of these two operations, compared to their optimal costs (full information available for free)?

The Model (3)

- Consider mixed sequence σ of Move and Find operations
- F(σ): subsequence of all Find operations in σ.
- M(σ): subsequence of all Move operations in σ.
- Cost(π): Sum of communication costs of all message transmissions performed during execution of protocol π.
- **Find-stretch** of D: $Stretch_{find}(\sigma) = \frac{Cost(F(\sigma))}{Opt\_Cost(F(\sigma))}$
- **Move-stretch** of D: $Stretch_{move}(\sigma) = \frac{Cost(M(\sigma))}{Opt\_Cost(M(\sigma))}$

The Main Result

The hierarchical directory server D guarantees:

- $Stretch_{find}$ = $O(\log^2 n)$ — $O(Deg_{read} * Rad_{read})$
- $Stretch_{move}$ = $O(\delta * \log n + \delta^2 / \log n)$ — $O(Deg_{write} * Rad_{write} * \delta + \delta^2 / \log n)$
- Memory requirement [bits] = $O(N * \delta * \log n + N * \delta^2 + n * \delta * \log^2 n)$

throughout the network, for handling N users, where $\delta = \lceil \log D(G) \rceil$.

D(G) = Diameter

Is this good or bad?

- Communication overhead of this tracking mechanism is within a polylogarithmic factor of the lower bound.
- Guaranteed overheads which are **polylogarithmic** in the **size** and **diameter** of the network.
- Main strength compared to conventional schemes: consideration of **locality** of Find and Move operations.

3. A Hierarchical, Distributed Directory Structure

A Distributed Data Structure

- Stores pointers to locations of each user in various nodes
- Pointers need to be updated as users move
- Allow some pointers to be inaccurate

„Intuitively, pointers at locations nearby to the user, whose update by the user is relatively cheap , are required to be more accurate, whereas pointers at distant locations are updated less often."

The Hierarchical Directory Server D

- composed of a hierarchy of $\delta = \lceil \log D(G) \rceil$ **regional directories RD_i** $(1 \le i \le \delta)$
- RD_i at level i of the hierarchy enables a searcher to track any user ξ within distance 2^i from it.
- The address stored for user ξ at RD_i is called the user's i'th level regional address: **$R\_addr_i(\xi)$.** It stores the address where ξ is currently expected to be.

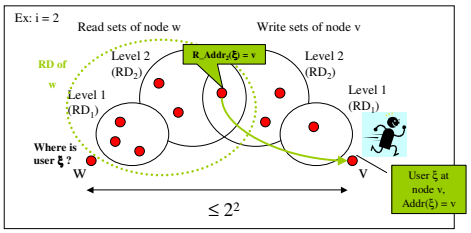
Implementation of a Regional Directory RD_i

Based on two sets:

- **$Write_i(v)$**: A node v reports every user it hosts to all nodes in some specified write set, $Write_i(v)$
- **$Read_i(w)$**: A searching node w queries all nodes in some specified read set, $Read_i(w)$

2^i-Regional Matchings

- $Read_i(w)$ and $Write_i(v)$ are guaranteed to intersect whenever v and w are within distance 2^i of each other.



2^i-Regional Matchings (2)

- Find operation invoked at node w is guaranteed to succeed only if dist(w,R_Addr(ξ)) ≤ m. $\left(m \le 2^{\lceil \log D(G) \rceil}\right)$
- Highest level RD_δ always succeeds!

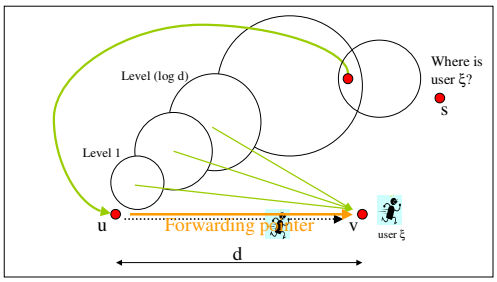
2^i-regional matching
$2^\delta = 2^{\lceil \log D(G) \rceil} \ge D(G)$

- What if diameter grows?
- What if R_Addr(ξ) is out of date?
  R_Addr(ξ) ≠ Addr(ξ)

„Forwarding Addresses“

- Whenever user ξ moves, it should update its regional directory RD_i **on all levels**.
- Too expensive!
  Use „Forwarding Addresses/Pointers“
- Update only ***log d*** lowest levels
  → the lower the level, the more up-to-date is the regional address.
- Low communication complexity, but
- $R\_Addr_i(\xi)$ might now point to an old address of ξ

„Forwarding Addresses“ (2)



„Forwarding Addresses“ (3)

- A forwarding address points at some more recent address of user ξ.
- But the user might have moved further!
- Define $\overline{A}(\xi) = \langle R\_Addr_1(\xi), \ldots, R\_Addr_\delta(\xi) \rangle$
  as the tuple of regional addresses of user ξ.
- $R\_Addr_1(\xi)$ = Addr(ξ)

The Reachability Invariant

The tuple of regional addresses $\overline{A}(\xi)$ satisfies the reachability invariant if for every level 1 ≤ i ≤ δ, at any time, $R\_Addr_i(\xi)$ stores a pointer Forward(ξ) pointing to the vertex $R\_Addr_{i-1}(\xi)$.

The proximity invariant

- How can **long chains of forwarding pointers** be avoided?
- Need **update mechanism** which updates the regional addresses frequently enough
- **$Migrate_i(\xi)$**:
  actual migration path traversed by ξ in its migration from $R\_Addr_i(\xi)$ to its current location, Addr(ξ).

The proximity invariant (2)

The regional addresses R_Addr$_i$(ξ) satisfy the proximity invariant if for every level $1 \le i \le \delta$, at any time, the distance travelled by ξ since the last time R_Addr$_i$(ξ) was updated in RD$_i$ satisfies

$$|Migrate_i(\xi)| \le 2^{i-1} - 1$$

The proximity invariant (3)

A node currently hosting user ξ has to maintain two data structures in order to guarantee the invariant:

- **Tuple of regional addresses**: $\overline{A}(\xi)$
- **Tuple of migration counters**: $\overline{C}(\xi) = \langle C_1(\xi), \ldots, C_\delta(\xi) \rangle$

Each $C_i(\xi)$ counts distance travelled by ξ since last update of R_Addr$_i$(ξ)

→ Enables to decide which regional addresses need to be updated after each move of ξ.

Updating regional addresses

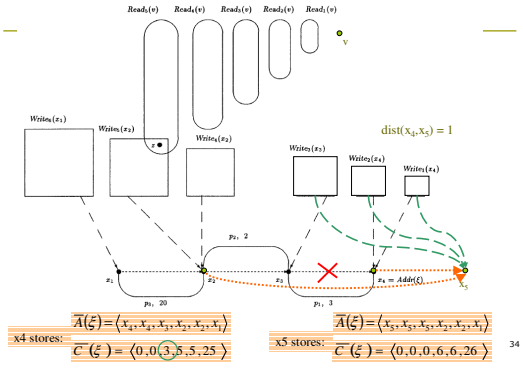
Whenever user moves from a node s to a node t:

1. Increase all migration counters C_i by dist(s,t).

If the highest level counter C_j reaches its upper limit (2^{J-1}-1): **Proximity invariant**

2. Update the regional directory at levels 1 to J: Set R_Addr(ξ) = t.
3. Set forwarding pointer at R_Addr$_{J+1}$(ξ) leading to t.
4. Relocate user ξ together with its tuples $\overline{A}(\xi)$ and $\overline{C}(\xi)$

Example



How are regional matchings constructed after all?

- Based on the concept of **sparse graph covers:** cover a graph by low-radius clusters with little overlap
- A matching can be constructed out of a cover with same degree and radius.
- The construction of a hierarchy of regional matchings amounts to a **global communication cost of O(E log^{4}n)**

Construction of a regional matching:

1. Start with m-neighbourhood cover $N_m(V)$ of the graph
2. Construct a coarsening cover C for $N_m(V)$
3. Select a centre l(T) in each cluster $T \in C$.
4. Select for every node v one cluster T_v and set Write(v) = {l(T_v)} and Read(v) = {l(T) | $v \in T$}

Important parameters of a regional matching

- **radius**: maximal distance from a node to any other node in its read or write set.
- **degree**: maximal number of vertices in any read or write set.

- Communication overhead of performing find and move operations in a RD$_i$ grows as the product of the degree and the radius of the related 2^i-regional matching.
- Trade-off between these two parameters (simultaneous minimization of radius and degree are a nontrivial task)

4. Handling Concurrent Accesses

Handling concurrent accesses

- Move and Find operations take some time.
- One cannot assume that there is enough time for system to complete an operation before getting a new request.
- What if someone attempts to contact a user **while he is moving**?
- What if this user **repeatedly moves** while he is being searched?

Modifications in the model

In the concurrent case, the two stages of the Find procedure cause problems:

- First stage: Retrieval of regional address **search for address can fail**
- Second stage: Tracing the user **„endless chases"**

Changing the Find operation

- Have to guarantee the retrieval of $R\_Addr_i(\xi)$ of user ξ even while this address is being changed.

Is there a problem at all?

A typical „atomicity problem" of asynchronous systems

Example scenario:

- Consider node v invoking a $Find(\xi,v)$ while $R\_Addr_i(\xi)$ is changed from s` to t`.
- Assume read requests from v to nodes $u \in Read_i(v) \cap Write_i(s`)$ to be very slow and hence reach u only after it has erased its pointer to s`.
- Assume read requests from v to nodes $u \in Read_i(v) \cap Write_i(t`)$ to be very fast and hence reach u before it made a pointer to t`.

⇒ **All read operations of node v fail to detect a pointer to ξ!**

Possible solution

Strengthen the definition of m-regional matchings by an additional requirement:
For every v and u, if $dist(v,u) \leq m$ then

$Read(v) \subseteq Write(u)$

Every node in Read(v) that pointed to s` is guaranteed to **point to either s` or t` at any time** during the move operation.

Preventing endless chases along forwarding pointers

- Idea: Searcher is allowed to „miss" the user while searching for him on level i only if the user is currently on transition to a new location farther away than distance 2^i.

This implies:

1. Searcher should retrieve a (old) regional address of the user at level i.
2. This address should allow a „short" chase to the user.

43

Preventing endless chases along forwarding pointers (2)

The **„clean move" requirement**:

- User ξ is not allowed to finish a new move which involves updating regional directories up to level *l*, before it is found by a searcher that has already received some $R\_Addr_i(\xi)$, for $i \leq l + 1$.
- Can be implemented with a **locking mechanism**.

44

General problems

- Insertion/Deletion of a new user results in high **initialization costs** of O(D(G) log n). **$O(D(G)*Rad_{write}*Deg_{write})$**
- Presented paper assumes a **static network** which **does not tolerate failures**
  → a mechanism is necessary to adapt the tracking structure to current topology.
- Proposed architecture is **relatively complex**

45

Experimental results

- Over 800 randomly generated networks of sizes 16 to 1024 nodes were analyzed.
- Read set size much smaller than theoretical upper bound of 4 $\log_2 n$.
  - **Linear networks**:
    read set size at any level at most 3.
  - **Grid networks** (deg 4, equal distances):
    max. read set size achievable is 9.

  ⇒ Communication complexities:
  FIND: $O(1)$
  MOVE: $O(\delta + \delta^2 / \log_{\delta} n)$

46

Summary

- Proposed strategy based on hierarchy of regional directories → **m-regional matchings**
- Main strength: considers **locality** of FIND and MOVES
- Communication overhead **within a polylogarithmic factor of the lower bound**.
- Alternative to centralized approaches used in **cellular networks**?
- Not suitable for architectures with no preexisting fixed-network infrastructure where node-connectivity is sporadic and databases are unstable (**ad-hoc networks)**.
  → **quorum systems**

47

Questions and Discussion

48