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
- 2. Model and Main Results of the Paper
- 3. A Hierarchical, Distributed Directory Structure
- 4. Handling Concurrent Accesses
- 5. Questions and Discussion

1. The Location Problem

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?

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

### 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

### 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

  overhead

  simplicity

  latency

### “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.

### “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 property:** exploit the fact that many moves are local

### 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.

### 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.
Organization of a Regional Directory

Organization based on a \( m \)-regional matching:
- Collection of sets of nodes, consisting of a read set \( \text{Read}(v) \) and a write set \( \text{Write}(v) \) for every \( v \in V \).
- The collection \( RW \) of all pairs of read and write sets,
  \( RW = \{ \text{Read}(v), \text{Write}(v) \mid v \in V \} \),
  is an \( m \)-regional matching, if
  \[ \text{Write}(v) \cap \text{Read}(u) \neq \emptyset \]
  for all \( v, u \in V \), s.t. \( \text{dist}(u, v) \leq m \).

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
- \( \text{dist}(u, w) \): length (weight) of shortest path between \( u \) and \( w \)
- \( \text{diameter}(G) \): maximal distance between any two vertices in \( G \)
- Operations are performed concurrently and asynchronously

The Model (2)

- \( \text{Addr}(\xi) \): current address of user \( \xi \)
- \( \text{Directory Server } D \): distributed data structure supporting the operations:
  - \( \text{Find}(\xi, v) \)
  - \( \text{Move}(\xi, s, t) \)

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 \( \sigma \) of Move and Find operations
- \( P(\sigma) \): subsequence of all Find operations in \( \sigma \)
- \( M(\sigma) \): subsequence of all Move operations in \( \sigma \)
- \( \text{Cost}(\sigma) \): Sum of communication costs of all message transmissions performed during execution of protocol \( \pi \)

- \( \text{Find-stretch} \) of \( D \):
  \begin{align*}
  \text{Find-stretch}(\sigma) &= \text{Cost}(\sigma) \\
  \text{Oper. Cost}(\sigma) &= \text{Find-stretch}(\sigma)
  \end{align*}

- \( \text{Move-stretch} \) of \( D \):
  \begin{align*}
  \text{Move-stretch}(\sigma) &= \text{Cost}(\sigma) \\
  \text{Oper. Cost}(\sigma) &= \text{Move-stretch}(\sigma)
  \end{align*}

The Main Result

The hierarchical directory server \( D \) guarantees:
- \( \text{Stretch}_{\text{Find}} = O(\log n) \)
- \( \text{Stretch}_{\text{Move}} = O(\delta \log n + B / \log n) \)
- Memory requirement [bits] = \( O(N^2 \delta^2 \log n + N^2 \delta^2 \log n) \) throughout the network, for handling \( N \) users, where \( B = \log 2\delta \).

\( \text{Init} = \text{Memory} \)
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.

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 regional directories $R_D_i$ ($1 \leq i \leq \delta$)

- $R_D_i$ at level $i$ of the hierarchy enables a searcher to track any user $\xi$ within distance $2^i$ from it.

- The address stored for user $\xi$ at $R_D_i$ is called the user’s $i$th level regional address: $R_{addr_i}(\xi)$. It stores the address where $\xi$ is currently expected to be.

Implementation of a Regional Directory $R_D_i$

Based on two sets:

- **Write($v$)**:
  A node $v$ reports every user it hosts to all nodes in some specified write set, Write($v$)

- **Read($w$)**:
  A searching node $w$ queries all nodes in some specified read set, Read($w$)

$2^i$-Regional Matchings

- Read($w$) and Write($v$) are guaranteed to intersect whenever $v$ and $w$ are within distance $2^i$ of each other.
2^\text{-Regional Matchings (2)}

- Find operation invoked at node \( w \) is guaranteed to succeed only if 
  \( \text{dist}(w, R_{\text{Addr}}(\xi)) \leq m \).
- Highest level \( RD_0 \) always succeeds!

\[
2^{f} = 2^{\log_{2}(D(G))} \\
\text{2^\text{-regional matching}}
\]

- What if diameter grows?
- What if \( R_{\text{Addr}}(\xi) \) is out of date?
  \( R_{\text{Addr}}(\xi) \neq \text{Addr}(\xi) \)

"Forwarding Addresses"

- Whenever user \( \xi \) moves, it should update its regional directory \( RD \), on all levels.
- Too expensive!
  Use "Forwarding Addresses/Pointers"
- Update only \( \log d \) lowest levels
  \( \xi \) the lower the level, the more up-to-date is the regional address.
- Low communication complexity, but
  \( R_{\text{Addr}}(\xi) \) might now point to an old address of \( \xi \)

"Forwarding Addresses" (2)

\[ \text{Where is user } \xi? \]

\[ \text{R_{Addr}(\xi) is pointed to earlier} \]

\[ \text{level 1 (Log d)} \]

\[ \text{level 0} \]

\[ \text{level 1} \]

\[ \text{A forwarding pointer} \]

\[ \text{R_{Addr}(\xi) as the region of addresses of user } \xi \]

"Forwarding Addresses" (3)

- A forwarding address points at some more recent address of user \( \xi \).
- But the user might have moved further!

- Define \( \overline{\text{Addr}}(\xi) = (R_{\text{Addr}}(\xi), ..., R_{\text{Addr}}(\xi)) \)

The Reachability Invariant

The tuple of regional addresses \( \overline{\text{Addr}}(\xi) \)
  satisfies the reachability invariant if for every level \( 1 \leq i \leq \delta \), at any time,
  \( R_{\text{Addr}}(\xi) \) stores a pointer \( \text{Forward}(\xi) \)
  pointing to the vertex \( R_{\text{Addr}}(\xi) \).

The proximity invariant

- How can long chains of forwarding pointers be avoided?
- Need update mechanism which updates the regional addresses frequently enough

- \( \text{Migrate}(\xi) \):
  actual migration path traversed by \( \xi \) in its migration from \( R_{\text{Addr}}(\xi) \) to its current location, \( \text{Addr}(\xi) \).
The proximity invariant (2)

The regional addresses $R_{\text{Addr}}(\xi)$ satisfy the proximity invariant if for every level $1 \leq i \leq \delta$, at any time, the distance travelled by $\xi$ since the last time $R_{\text{Addr}}(\xi)$ was updated in RD, satisfies

$|\text{Migrate}(\xi)| \leq 2^{i-1} - 1$

Example

![Example Diagram]

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 $O(E \log n)$

Construction of a regional matching:
1. Start with $m$-neighbourhood cover $N_m(V)$ of the graph
2. Construct a coarsening $C$ 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

$\text{Write}(v) = \langle l(T_v) \rangle$ and $\text{Read}(v) = \langle l(T) \mid v \in T \rangle$

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, 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)

Updating regional addresses

Whenever user moves from a node $s$ to a node $t$:
1. Increase all migration counters $C_i$ by $\text{dist}(s,t)$.
2. Update the regional directory at levels $1$ to $J$:
   Set $R_{\text{Addr}}(\xi) = t$.
3. Set forwarding pointer at $R_{\text{Addr}}(\xi)$ leading to $t$.
4. Relocate user $\xi$ together with its tuples $\pi(\xi)$ and $\tau(\xi)$.

The proximity invariant (3)

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

- Tuple of regional addresses: $\pi(\xi)$
- Tuple of migration counters: $\tau(\xi) = (c_1, c_2, \ldots, c_J)$

Each $c_i(\xi)$ counts distance travelled by $\xi$ since last update of $R_{\text{Addr}}(\xi)$

$\rightarrow$ Enables to decide which regional addresses need to be updated after each move of $\xi$. 

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(ξ) 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(ξ,v) while R_Addr(ξ) is changed from s’ to t’.
- Assume read requests from v to nodes u ∈ Read(v) ∩ Write(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 ∈ Read(v) ∩ Write(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) ≤ m then

\[
\text{Read(v)} \subseteq \text{Write(u)}
\]

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 2i.

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.

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(ξ), for i ≤ l + 1.
- Can be implemented with a locking mechanism.

General problems

- Insertion/Deletion of a new user results in high initialization costs of O(D(G) log n).
- 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

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 log2n.
  - 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: \(O(\log_2 n)\)

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

Questions and Discussion