Data-Centric Query in Sensor Networks II

Jie Gao
Computer Science Department
Stony Brook University
Double Rulings: extend GHT and rumor routing

• Hash data to a 1-d curve, instead of a 0-d point
• Motivations for generalization
  – Data delivery uses multi-hop routing
    • Leave information along route at no extra cost
  – More flexible data retrieval
    • Easier to encounter a 1-d curve than a 0-d point
Rectilinear Double Ruling

- Rectilinear Double Ruling
  - Producer stores data on horizontal lines
  - Consumer searches along vertical lines
  - Correctness: Every horizontal line intersects every vertical line
  - Distance sensitive: q finds p in time $O(d)$, where $d = |pq|$. 
Spherical Double Rulings Scheme

- Producer follows a circle to the hashed location
  - Includes GHT as a sub-case
  - Allows a large variety of retrieval mechanisms

- Improves on GHT
  - Load balancing for popular data types
  - Distance sensitivity
  - Flexible data retrieval schemes improve system robustness
Double Rulings on a Sphere

• Stereographic projection maps a projective plane to a sphere
  – Circles map to circles
  – May incur distortion

• For a finite sensor field
  – Can choose location and size of sphere such that distance distortion is bounded by $1+\varepsilon$. 
Spherical Double Rulings

- Any two great circles intersect
  - Use great circles in place of vertical/horizontal lines
Spherical Double Rulings

• One major difference with rectilinear double rulings:
  – **Infinitely** many great circles through a point
  – A lot more flexibility
Data Replication

- Data centric hash function $h(T_i) = h_i$.
- Producer $p$ replicates data along the great circle $C(p, h_i)$.
Data Replication

• Different producers with the same data type hash to different great circles, all passing through \( h \), and its antipodal point \( \bar{h} \).
  – Allow aggregation.
Replication Curve Examples

Hashed node

Antipode

Producer 1

Producer 2

Replication curve

GHT paths
Data Retrieval

• Flexible retrieval rules
  1. GHT Style Retrieval
  2. Distance Sensitive Retrieval
  3. Aggregated Data Retrieval
  4. Full Power Data Retrieval
1. GHT Style Retrieval

• GHT still works
• Consumer $q$ wants data $T_i$

Consumer goes to hashed node $h$ or its antipodal, whichever is closer.
2. Distance Sensitive Retrieval

- Distance Sensitive: If producer is at distance $d$ from $q$, consumer should find data with cost $O(d)$.
  - Consumes less network resources
  - Users are likely to be more interested in immediate vicinity.
  - Lower delay --- Important in emergency response.
2. Distance Sensitive Retrieval

- Rotate the sphere so that hashed node is at the north pole.

If $q$ is $d$ away from $p$, the distance from $q$ along latitude curve is $\leq d \cdot \pi/2$. 
2. Distance Sensitive Retrieval

- Distance Sensitive: If producer is at distance \(d\) from \(q\), consumer should find data with cost \(O(d)\).

Consumer \(q\) follows the circle with fixed distance to the hashed location.

- Wrong direction?
  - Handled using a doubling technique
  - A random choice of direction works well in practice (we use this in simulations).
2. Distance Sensitive Retrieval

![Distance Sensitive Retrieval Diagram]

- **Hashed node**: Node used for hashing in distance-sensitive retrieval.
- **Antipode**: The antipode is a concept in distance-sensitive retrieval, often used to define a boundary or limit.
- **Producer**: The producer is the entity generating or providing data.
- **Consumer**: The consumer is the entity receiving or utilizing the data.
- **Retrieval curve**: The retrieval curve represents the path or strategy used for retrieving data within a certain distance.
3. Aggregated Data Retrieval

- Consumer wants data of several Data Types \( \{T_i\} \)
  - E.g., monkey & elephant detections.
  - Correctness: Any closed cycle that separates \( h_i \) from its antipodal intersects the producer curve.
  - Many such retrieval curves! \( \Rightarrow \) more freedom for consumers and better load balancing.
3. Aggregate Data Retrieval

- Hashed node
- Antipode
- Producer
- Consumer
- Retrieval curve
4. Full Power Data Retrieval

- Consumer wants all the data in the network

Follow a great circle, retrieve all data.

- Correctness: Any two great circles intersect
- Many such great circles!
4. Full Power Data Retrieval

- Producer
- Antipode
- Hashed node
- Great Circle Retrieval curve
- Consumer
Local Data Recovery upon Node Failures

- When a group of nodes are destroyed,

All the data on those nodes are available on the boundary of destroyed region.
Local Data Recovery upon Node Failures

Survived Data Replicas on the boundary
Difference of spherical v.s. Rectilinear double rulings?

- All lines are great circles that pass through the point of infinity.
- The point of infinity is the hash location!
Implementation

• How to forward data on a virtual curve?
  – Use “Geographic Greedy forwarding on a Curve”

• The question of density
  – Is it always possible to forward?
  – Simulation: A suitable 2-hop neighbor exists with high probability, for networks with avg degree ≥5.
Simulation: Distance Sensitivity

Distance Sensitivity of queries

- GHT
- GLIDER scheme
- Spherical Double Ruling

4200 nodes with average degree 8 per node.

Distance Sensitivity of queries

INFOCOM 2006.

Simulation: Storage/Retrieval Tradeoff

Nodes on replication curve can store the data or a pointer to the actual data.

- Larger Replication Interval
- Decreasing Storage Cost
- Increasing Consumer Cost

![Graph showing the tradeoff between replication interval and storage/retrieval costs]
Simulation: Storage/Retrieval Tradeoff

More storage, Lower retrieval cost.

Replication only on the hashed node and antipode.
Simulation: Load Balancing

500 consumers querying a popular data item

Double Ruling

GHT

Load Distribution

Jie Gao, CSE595-spring09
Discussion

• Data collection by mobile data mules.
  - Physically move along any retrieval curve.

• Advanced hashing schemes.
  - E.g., similar data types are placed nearby.

• Networks with holes.
  - Require special care.
When sensors are not regular...

• Double rulings on an irregular shape.
  – Shape parameterization

• Integrate double rulings with other approaches.
Two-level brokerage structure

- Recall GLIDER: landmark-based routing.

- Partition the sensor field into tiles.
  - GHT on the tiles.
  - Double rulings inside each tile.
At the CDT level: GHT on tiles

- Hashing on coarse data types for structured data storage.

• Both producers and consumers of the same content type follow the shortest path tree to the hashed tile (the root of the tree).
• Consumers return once the data are retrieved, otherwise move on towards the hashed tile.

Large-sized Animals
- giraffes
- elephants
- ……

DHT at a coarse data type level
Stored in the same tile
Within Each Tile – Double-ruling in Transit Tiles

Guides $v$, $x$, $y$ are landmarks selected according to a set of rules based on hashing and the CDT

An example by simulation

- Routes formed by following shortest paths to guides
- The two sets of curves always meet
Double-ruling in Hashed Tile
Reducing Consumer Query Cost – Simpler Retrieval Route

Case I: meet in tile $u$

Case II: will meet in tile $x$. But this shouldn’t happen if we follow shortest path on CDT.
Load Balancing within Each Tile

- Multiple producer routes inside a tile.
- A consumer can pick up everything from a tile.
Discussion on double rulings

- Design for irregular shaped sensor field.
- Landmark-based double rulings
  - Example: 2 landmarks, red curve – $d_1+d_2=\text{const}$, blue curve – $|d_1-d_2|=\text{const}$
- Use sensor data
  - Gradient
  - Iso-contours
Location service


Location service

- Geographical routing requires obtaining the location of the destination.

- What if the sensors move? How to update the location information?

- Internet: domain name server (DNS) translates user-friendly domain name (www.cnn.com) to machine-friendly IP address.
Centralized v.s. distributed location service

• Location server stores the mapping between locations and node IDs.
  – Centralized approach, single point of failure.
  – Communication bottleneck.
  – Location server might be far away.

• Distributed location servers: every node participates and acts as location servers for others.
Challenges

- Problem 1: each node need to know the location server of any node.
  - To update its own location info upon movement.
  - Query for the location of any other node.
- Problem 2: how to get to the location server?
  - We need a routing algorithm, say geographical routing.
- Problem 3: geographical routing requires the knowledge of destinations.
  - How to get the location of the location server?
  - Every node can be moving.
- Problem 4: location update upon node movement.
Proposal 1: Use GHT

• Each node holds a data = its location
• Use GHT to store the data by a hash function.

• Problem:
  – Distance insensitivity.
  – Frequent location update upon node movement.
Proposal 2: use double rulings

- Each node stores the data at the nodes on the producer curve.

- Problem: cost is too high for mobile nodes.
  - Producer curve has length $\sim \sqrt{n}$.
Locate a mobile user

- **“Move”** operation:
  - inform system of new location

- **“Find”** operation:
  - Locate user at his current address.

- Distance-sensitivity: move to nearby locations or search for nearby users should be cheap.
  - Most moves are local
  - Most queries are local, too.
Model

- Connected, undirected, weighted graph $G$.
- Weight $w(e)$: cost on edge $e$.
- $\text{dist}(u, v)$: length of shortest path.
- Diameter $D(G)$: max distance of any two nodes in $G$.
- Address $\text{Addr}(x)$: current location of $x$.
- Assume an efficient routing scheme.
Model

• Consider a mixed sequence $\sigma$ of Move and Find operations.
  – $F(\sigma)$: subsequence of all Find operations in $\sigma$.
  – $M(\sigma)$: subsequence of all Move operations in $\sigma$.

• Cost: message transmissions.

• **Find-stretch**: cost $\frac{F(\sigma)}{OPT(F(\sigma))}$

• **Move-stretch**: cost $\frac{M(\sigma)}{OPT(M(\sigma))}$

• Goal: make Find-stretch and Move-stretch polylogarithmic in $n$. 
A distributed data structure

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

“Pointers at locations nearby to the user, whose update by the user is relatively cheap, are required to be more accurate, whereas pointers at distance locations are updated less often.”
Hierarchical directory server

- A hierarchy of $\delta = \lceil \log D(G) \rceil$ regional directories $\text{RD}_i$ ($1 \leq i \leq \delta$)

- $\text{RD}_i$ at level $i$ of the hierarchy enables a searcher to track any user within distance $2^i$.

- The address stored for user $x$ at $\text{RD}_i$ is called $i$-th level regional address $\text{R_addr}_i(x)$ --- where $x$ is currently expected to be.
Regional directory $\text{RD}_i$

- **Write$_i(v)$**
  - A node $v$ reports every user it hosts to all nodes in the write set.

- **Read$_i(w)$**
  - A searching node $w$ queries all nodes in some read set.

- **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 matching

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

Ex: i = 2

Read sets of node w

Write sets of node v

Level 2 (RD_2)

Level 1 (RD_1)

R_Addr_2(ξ) = v

User ξ at node v, Addr(ξ) = v

\[ \leq 2^2 \]
2^i-regional matching

- Find operation invoked at node w will succeed at the lowest possible level enabled by the distance from w to v.
- At the highest level this operation will always succeed.

- Two questions:
  - What if the network diameter grows?
  - What if the address maintained at the location directory is out of date?
“Forwarding addresses”

• Whenever a user x moves, it update its regional directory on all levels.
  – Too expensive!

• Update only log d lowest levels
  – The lower the level, the more up-to-date is the regional address
  – Low communication cost
  – The address $R_{addr_i}(x)$ might point to an old address.
“Forwarding addresses”

Where is user $\xi$?

S

Forwarding pointer

user $\xi$

Level (log d)

Level 1

u

v

d

3/5/09

Jie Gao, CSE595-spring09
The reachability invariant

• Define the tuple of regional addresses

\[ A(x) = < R_{Addr_1}(x), \ldots, R_{Addr_\delta}(x) > \]

• \( R_{Addr_1}(x) = \text{Addr}(x) \) the true address.

• The reachability invariant: if at any time, \( R_{Addr_i}(x) \) stores a pointer \textbf{Forward} \( (x) \) to \( R_{Addr_{i-1}}(x) \).

• This may result in a long chain of forwarding pointers.
The proximity invariant

- The reachability invariant: if at any time, the distance travelled by x since the last time R_Addr_i(x) was updated satisfies
  $$|\text{Migrate}_i(x)| \leq 2^{i-1} - 1$$
- Migrate_i(x): the actual migration path from R_Addr_i(x) to its current location.
- A node currently hosting user x maintains a tuple of migration counters:
  - $C_i(x)$: distance travelled since the last update.
Updating regional addresses

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

- Increase all migration counters $C_i$ by $\text{dist}(s, t)$.

If the highest level counter $C_j$ reaches the upper limit ($2^{j-1} - 1$)

- Update the regional directory at levels 1 to $j$: set to $t$.
- Set forwarding pointer at $R_{\text{Addr}_{j+1}}(x)$ leading to $t$.
- Relocate user $x$ together with its tuples $A(x)$ and $C(x)$.
Example
Discussion

- Proof of the cost in the full paper.
- Location service for one mobile user.
- What if all the nodes in the network are mobile?