Data-Centric Query in Sensor Networks

Jie Gao
Computer Science Department
Stony Brook University
Papers

- **[Intanagonwiwat00]** Chalermek Intanagonwiwat, Ramesh Govindan and Deborah Estrin, *Directed diffusion: A scalable and robust communication paradigm for sensor networks*, MobiCOM '00. *The first paper on data-centric routing in sensor networks. Data discovery relies on flooding the network.*


- **[Sarkar06]** Rik Sarkar, Xianjin Zhu, Jie Gao, *Double Rulings for Information Brokerage in Sensor Networks*, MobiCom06. *Hash data to circles.*
Scenario I: tourists and animals

- A sensor network in a zoo.
- A tourist asks: where is the elephant?
- So which sensor has the data about the elephant?
Scenario II: location service

- A missing part of routing with geographical or virtual coordinates: how does the source know the location (or virtual coordinates) of the destination?

- Location service: a brokerage service that answers queries such as: where is the node with ID 23?

- Geographical routing:
  - The source asks for the location of destination;
  - The source routes by using geographical routing.

- Notice: chicken and egg problem.
Data-centric

• Traditional networks: routing is based on network ID (e.g., IP addresses).

• Sensor networks: communication abstractions are based on data rather than node network addresses.

• Data-centric routing
  – Route to the node with the data the user wants.

• Data-centric storage
  – Store/sort the data by data type (elephant).
Abstraction of data-centric routing

• Information producer/consumer problem.

• Information producer.
  – Can be anywhere in the network.
  – Dynamic, mobile.
  – Multiple producers generating data about the same data type.

• Users = information consumer.
  – Can be anywhere in the network.
  – Concurrent multiple consumers.
Challenges

- Information producers/consumers have no idea about each other.
- Yet we want them to find each other quickly.

- Main approaches:
  - **Push-based**: producers do most of the work.
  - **Pull-based**: consumers actively search.
  - **Push-pull**: both producers/consumers search to find each other.
This class

• Directed diffusion
  – Pull-based
• Geographical hash table
• Rumor routing
• Double rulings
  – Push-pull
  – In-network storage
Directed diffusion

• Data is named by **attribute-value pairs**.

  type = four-legged animal  // type of animal seen
  instance = elephant       // instance of this type
  location = [125, 220]     // node location
  intensity = 0.6           // signal amplitude measure
  confidence = 0.85         // confidence in the match
  timestamp = 01:20:40      // event generation time

• Query is represented by **interest**.

  type = four-legged animal  // detect animal location
  interval = 20 ms           // send back events every 20 ms
  duration = 10 seconds      // .. for the next 10 seconds
  rect = [-100, 100, 200, 400] // from sensors within rectangle
Interest dissemination

- A sensing task is disseminated in the network as an interest for named data.
- Interest is refreshed for robustness.
Gradient establishment

- Each node caches a gradient for interest: which specifies the data rate and duration.
Data transmission

- Data is transmitted back to sink.
- Multi-path can be adopted.
- Good paths (low delay, more reliable ones) are reinforced.
Pros and Cons

• The first scheme for data-centric routing.
• Pull-based approach.
• Ok for streaming data type – the cost for flooding is amortized.
• Flooding is expensive for infrequent queries, or queries that only involve a small set of nodes.
Distributed hash table (DHT)

- For Bob and Alice to find each other.
- “Lost and found”.

- Basic idea: data-dependent rendezvous.

- Use a content-based hash function
  \( h(\text{elephant}) = \text{sensor #10} \).
  - All the sensors with elephants info send to #10.
  - All the tourists interested in elephants go to #10 to fetch the information.
Distributed hash table (DHT)

- Originally proposed for Peer-to-Peer routing on the Internet.
  - E.g, Chord, Pastry, Tapestry, etc.

- A data object is given a key.
- Each node saves a set of keys.
- A routing algorithm allows any node to locate the one with an arbitrary key.
Geographical hash table (GHT)

- Assume nodes know their locations and do geo-routing.
- The content-based hash function outputs a geographical location: $h(\text{elephant}) = (14, 22)$.
- Use geographical routing for information producers/consumers to route to the rendezvous.
Geographical hash table (GHT)

- The content-based hash function
  \( h(\text{elephant}) = \text{a geographical location} \ (14, 22) \).

- Use geographical routing for information producers/consumers to route to the reservoir.

- Two questions:
  - What if there is no sensor at location \((14, 22)\)?
  - What if geographical routing gets stuck?
Geographical hash table (GHT)

- We route to location \( L = (14, 22) \) and geographical routing finds out there is no way to \((14, 22)\) by touring along a perimeter of a face and get back to where it started.

Home node: the one that is geographically closest to \( L \).

Home perimeter: the perimeter that geographical routing tours around.
Geographical hash table (GHT)

- We replicate elephant information on all the nodes on the perimeter.
- The query follows the same home perimeter and retrieve the message.

Home node: the one that is geographically closest to L.

Home perimeter: the perimeter that geographical routing tours around.
GHT: maintenance

- Home node periodically refresh replication by sending a packet to the hashed location L.
- If the timer of the replica times out, then a replica node initiates a refresh.
Hierarchical replication

• To reduce bottleneck at the hash nodes and improve data survivability under node failure
• Hash location is replicated at each level of a quad tree.
Geographical hash table (GHT)

- Advantages:
  - simple.
  - load balancing in storage.

- Disadvantages:
  - Not locality-sensitive. Consumer may travel far to fetch data even if the producer is close.
  - Fault tolerance?
  - Overload nodes on the boundary.
  - Nodes with popular data become bottleneck.
Rumor routing

- Producer: route along a line or random walk, and leave data traces on the way.

- Consumer: route along another line or random walk, hope to pick up the data.
A geometric observation

- Inside a circle, draw two random lines, what is the probability that they intersect?

\[
\int_0^1 x(1-x) \cdot 2dx = \frac{1}{3}
\]
A geometric observation

• Inside a circle, draw \( k \) random lines, what is the probability that another random line intersects at least one of the \( k \) lines?

\[
\Pr(k) = 1 - \left(1 - \frac{1}{3}\right)^k = 1 - \left(\frac{2}{3}\right)^k
\]

\( \Pr(5) = 87\% \)
\( \Pr(10) = 98\% \).

\( \Pr(\log n) = 1 - O(1/n) \).
Algorithm Basics

- All nodes maintain a neighbor list.
- Nodes also maintain a event table
  - When it observes an event, the event is added with distance 0.
- Agents
  - Packets that carry local event info across the network.
  - Aggregate events as they go.
- Agents do a random walk: among the 1-hop neighbors, find one that is not visited recently.
Examples
Simulation results

- $N=3000-5000$, randomly in 200 by 200 field, communication radius is 5. The diameter of the network is roughly 40.
- $A$: # agents, $La$=agent TTL, $Lq$=query TTL.
Some thought about simulation results

- Random walk is not necessarily straight.
- Random walk on a graph: move to a neighbor with probability $1/d$, where $d$ is the degree.
- **Hitting time** $H(i, j)$: expected number of steps to reach $j$ if we start from node $i$.
- Suppose the source is $i$, sink is $j$, then the total number of hops of the two random walk before they intersect $= H(i, j)$ approximately.
Some thought about simulation results

- For general graph the hitting time is $\Theta(n^3)$.
- For complete graph the hitting time is $O(n)$.
- The maximum hitting time between any two nodes is at least half of the expected number of steps before a random walk visits half of the nodes.
- So there are two nodes such that a random walk between them visits about $\Omega(n)$ nodes.

Random walk on graphs, a survey, by Lovasz.
Rumor routing

- Producer curve and consumer curve intersect with some probability.
- Random walk can be expensive.

- Idea: design producer curve and consumer curve such that they always intersect.
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 $1+\varepsilon$. 

$r$ $h^*$ $h$
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
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
3. Aggregated Data Retrieval

- Consumer wants data of several Data Types \( \{ T_i \} \)
  - E.g., monkey & elephant detections.

  Follow a closed curve that separates \( h_i \) and its antipodal point, for each data type \( T_i \)

  - 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
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 \( \geq 5 \).
Simulation: Distance Sensitivity

Distance Sensitivity of queries

Simulation: Storage/Retrieval Tradeoff

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

Increasing Consumer cost

Larger Replication Interval Decreasing Storage Cost
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

Load Distribution

GHT
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
Discussion

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

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