# Lecture 5:

# Data Query in Networks

Rik Sarkar

# Papers

Chalermek Intanagonwiwat, Ramesh Govindan and Deborah Estrin, **Directed diffusion: A scalable and robust communication paradigm for sensor networks**, MobiCom '00

Sylvia Ratnasamy, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, Brad Karp, Scott Shenker**, GHT: A Geographic Hash Table for Data-Centric Storage**, In First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA) 2002

David Braginsky, Deborah Estrin, **Rumor routing algorithm for sensor networks**, 1st ACM workshop on Wireless Sensor Networks, 2002

Rik Sarkar, Xianjin Zhu, Jie Gao, **Double Rulings for Information Brokerage in Sensor Networks**, MobiCom '06

# Problem: How to find the data?

- A tourist in a park asks
- “Where is the elephant?”
- Out of all the sensors/cameras which one is close to an elephant?

# Data centric routing

- Traditional networks try to route to an IP address
- Find path to the node with a particular ID
- But what if we try to find data, not specific nodes?
- After all, delivering data is the ultimate goal of routing and networks
- Data centric storage
  - Storage depends on the data (elephant, giraffe…)
- Data centric routing (search)
  - Route to the data

# Network as a Database

- Information Producer
  - Can be anywhere in the network
  - May be mobile
  - Many producers may generate data of the same type
- User or Information Consumer
  - Can be anywhere
  - May be many

# Network as a Database: Challenges

- Consumer does not know where the producer is, and vice versa
- Need to search : Must be fast, efficient

Basic methods:

- Push: Producer disseminates data
- Pull: Consumer looks for the data
- Push-pull: Both producer, consumer search for each-other

# Directed Diffusion

- Data has **attribute-value** pairs.

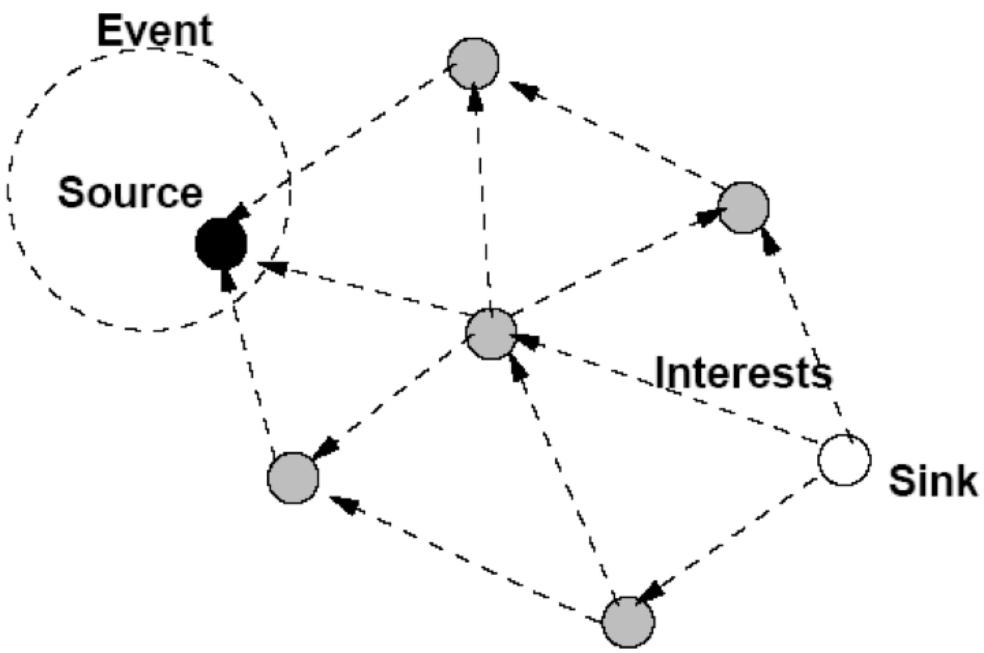
```
type = four-legged animal // type of animal seen
instance = elephant       // instance of this type
location = [125, 220]     // node location
intensity = 0.6           // signal amplitude measure
confidence = 0.85         // confidence in the match
timestamp = 01:20:40      // event generation time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

- Query Says what it is **interested** in:

```
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
```

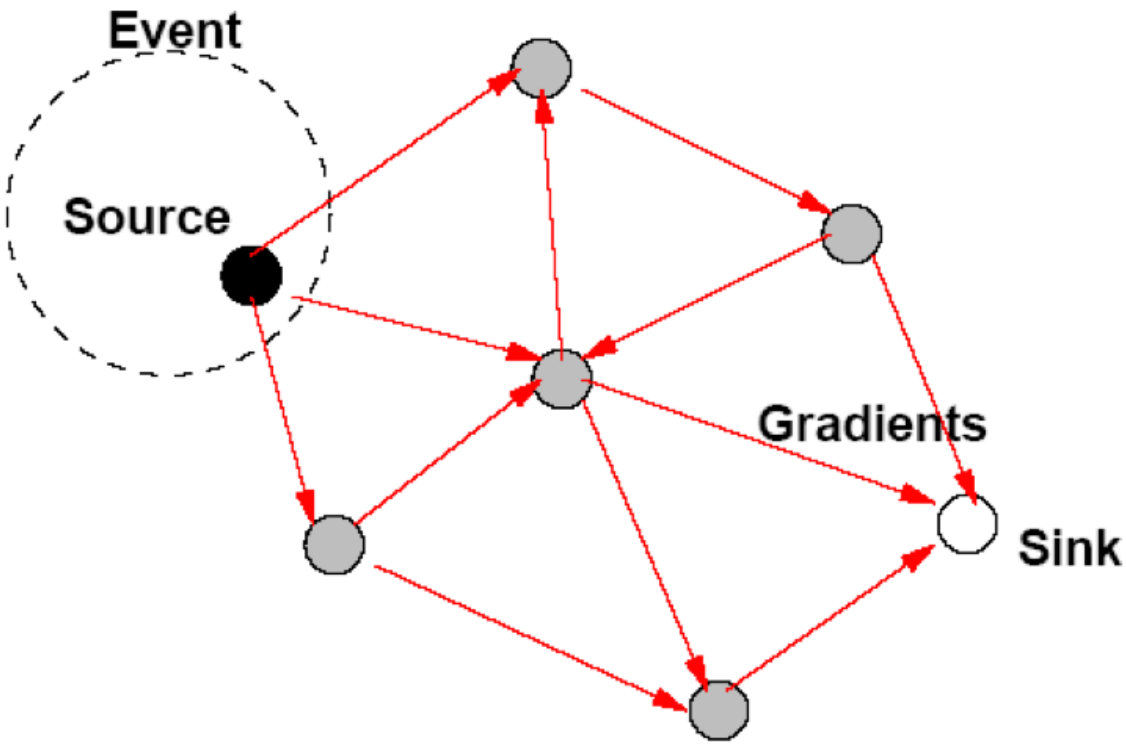
# Interest Dissemination

- Interest is disseminated in network
- Essentially flood a query



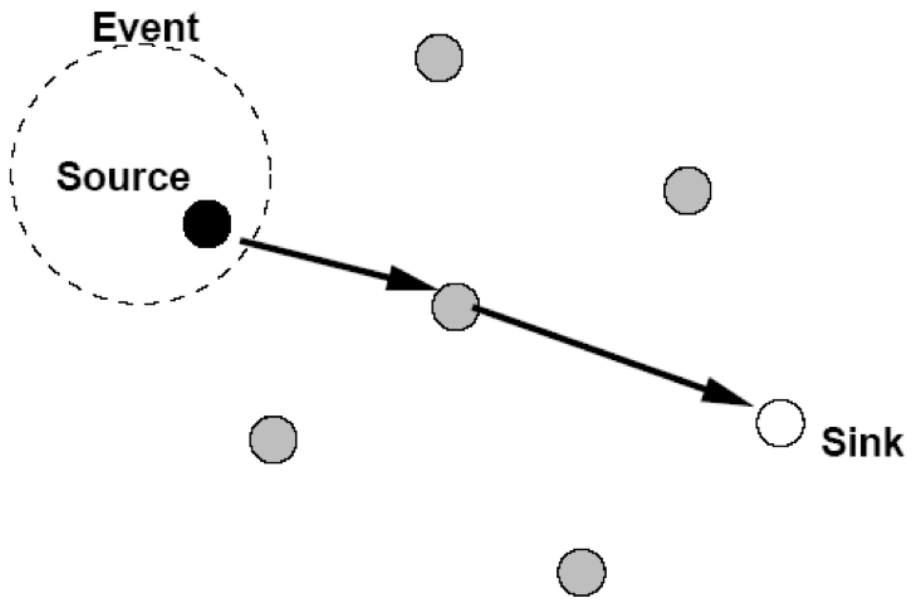
# Establish a Gradient

- Each node stores “gradient” of interest: data rate and diffusion



# Data return to sink or query node

- Along gradient paths
- Possibly multiple paths
- Preferably low delay paths



# Discussion

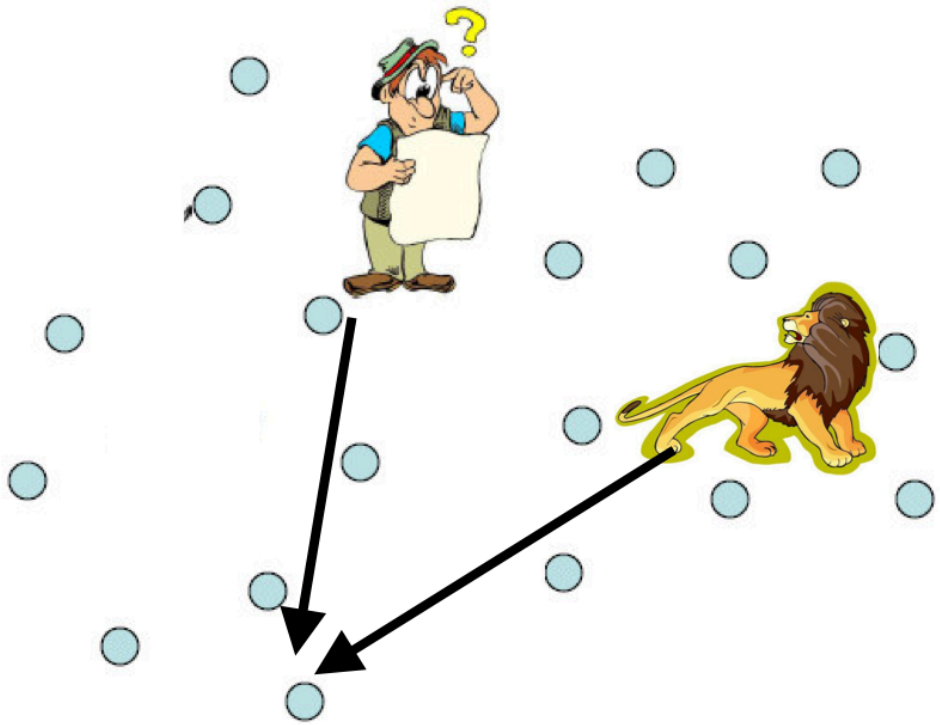
- Good for long term data gathering : Report to me all temperatures > 15 C from 1pm to 4pm
- There will be many such data anyway, so cost of one flooding is not a problem
- Bad for more specific isolated queries where there will be only a few replies

# Distributed hash tables

- Use a hash on the data: **h**(song1.mp3) = node#26
- Anyone that has song1.mp3 informs node#26
- Anyone that needs Song1.mp3 checks with node#26
- Used in peer to peer systems like Chord, pastry etc

# Geographic Hash Tables

- Content based hash gives coordinates:
  - **h**(lion) = (12, 07)
- Producer sends msg to (12, 07) by geographic routing and stores data
- Consumer sends msg to (12, 07) by geographic routing and gets data

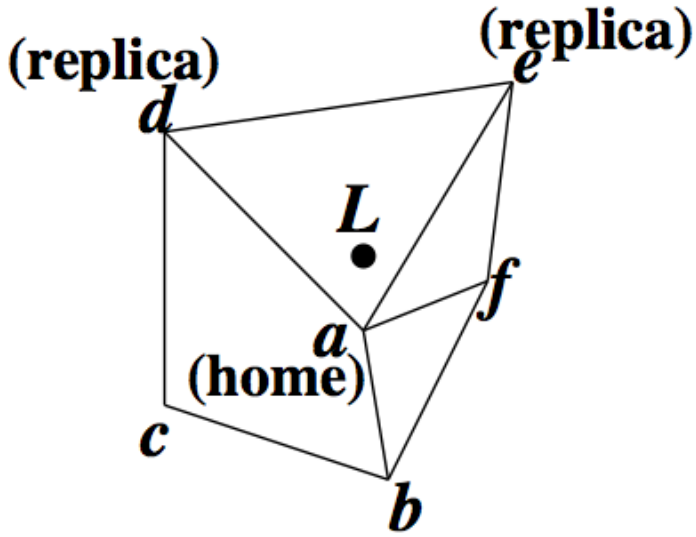


# GHT

- What if there is no sensor at (12, 07) ?
- What if geographic routing gets stuck before it gets to (12, 07) ?

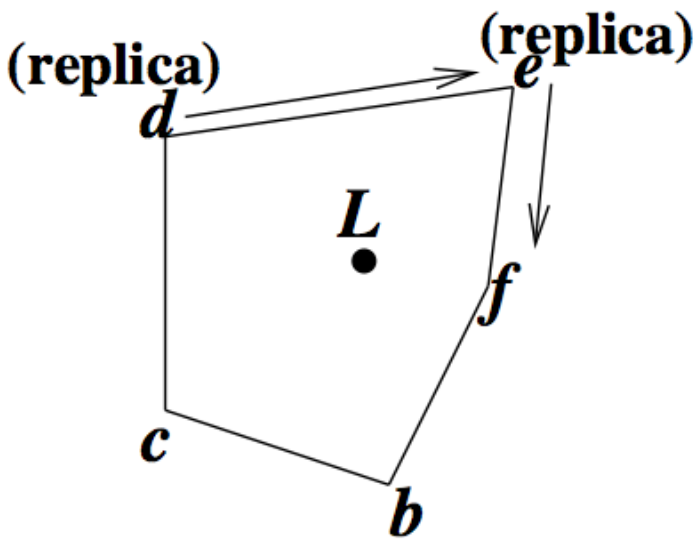
# GHT

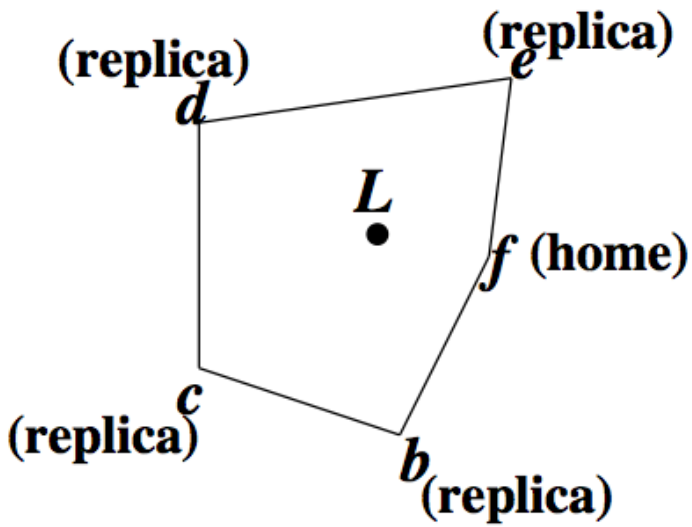
- L = hash location
- *ade* : face that contains L
- GHT stores copies of data on a,d,e
- a is in charge : home node : makes sure data is fresh, all nodes on perimeter has data



# Fault handling

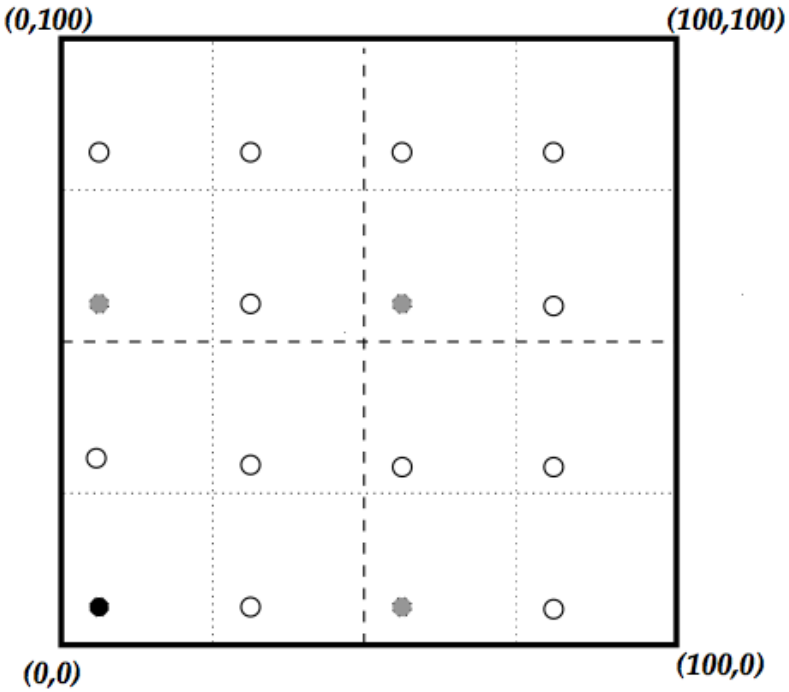
- What if home node a dies?
- Replicas have a timer that triggers a new check
- A new node becomes home



# Fault and load handling

- A few nodes have all the responsibility: too much load, susceptible to failure
- Hash location is replicated at each level of a quadtree down to some fixed depth



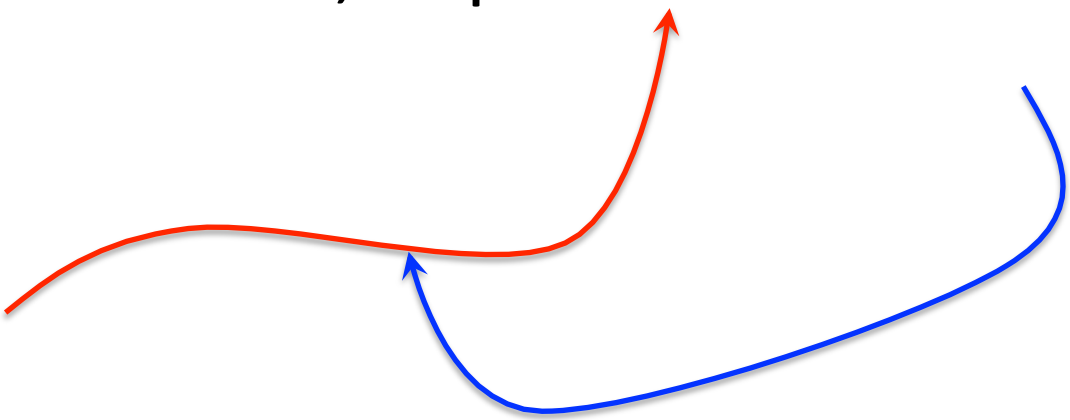
# GHT

- Advantages
  - Simple
  - Handles load balancing and faults
- Disadvantages
  - Not locality sensitive: everyone has to go to hash node even if producer and consumer are close
  - Overloads boundaries of holes
  - If a data is queried or updataed often, that node has a lot of traffic – bottleneck

# Rumor Routing

- Producer: Send data along a curve or random walk, leave data or pointers on nodes

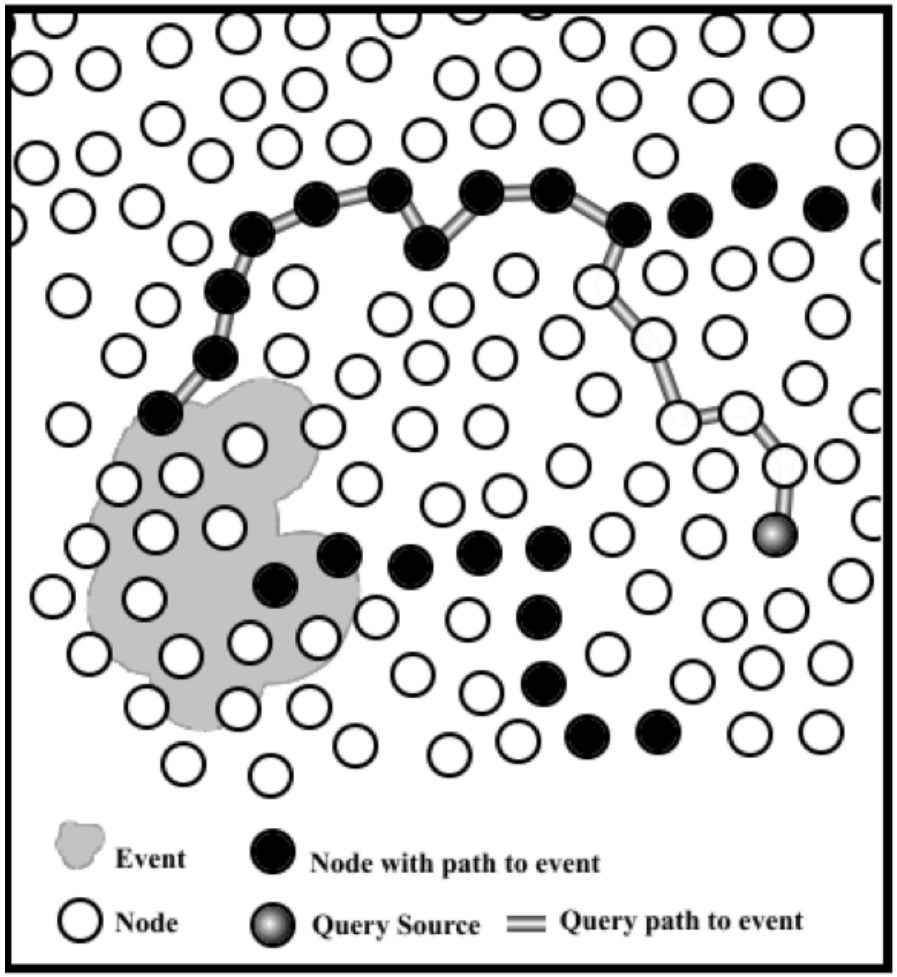
- Consumer: Route along another curve or random walk, hope to meet data or pointer

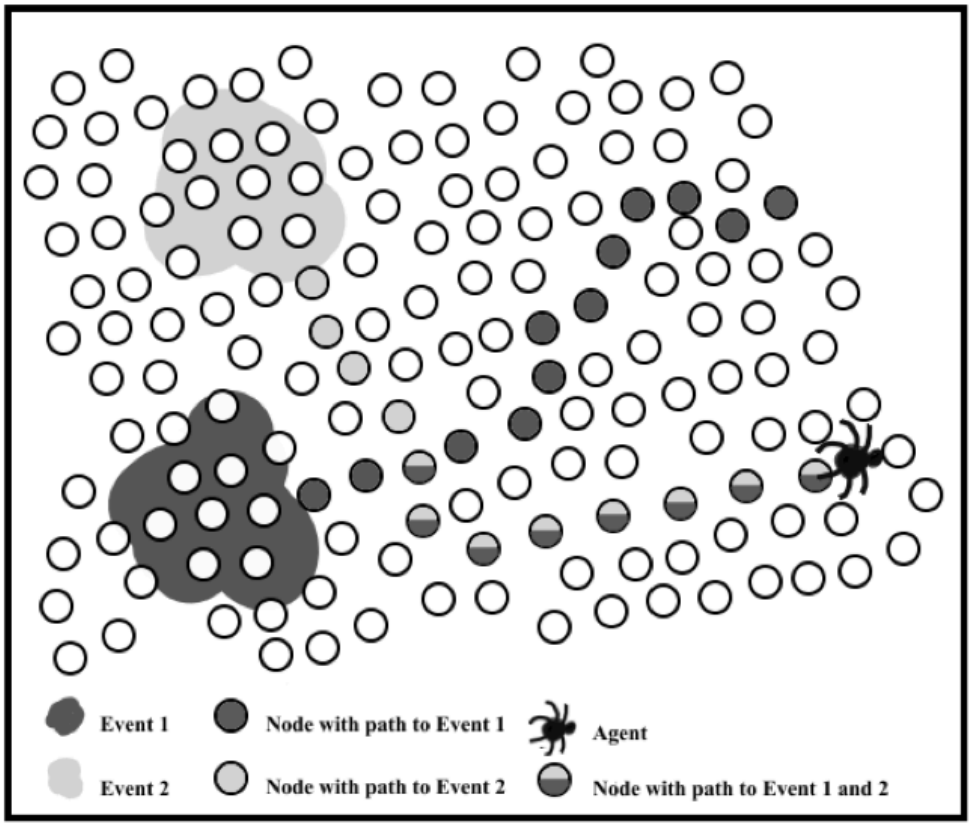


# Rumor routing

- Each node maintains a list of events
- Adds events as they happen

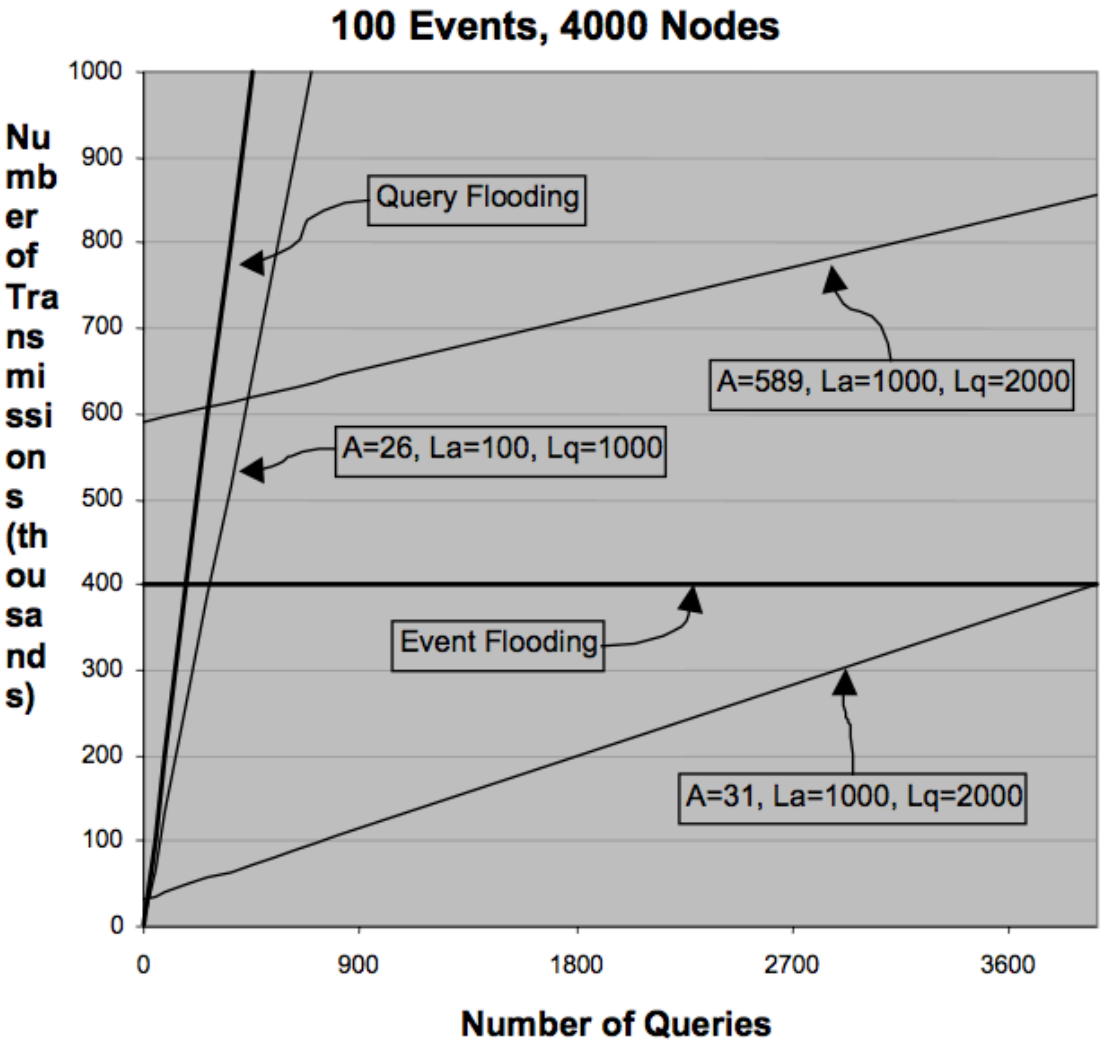
- Agents: Packets that carry events in the network
  - Aggregate events of each node they pass through
- Agents move in random walk. From 1-hop neighbors select one that has not been visited recently


Event
Node with path to event
Node
Query Source
Query path to event


Event 1
Node with path to Event 1
Agent
Event 2
Node with path to Event 2
Node with path to Event 1 and 2

# Simulation

- 200x200 field, communication radius 5
- A = #agents, La=agent TTL, Lq=query TTL



# Problems

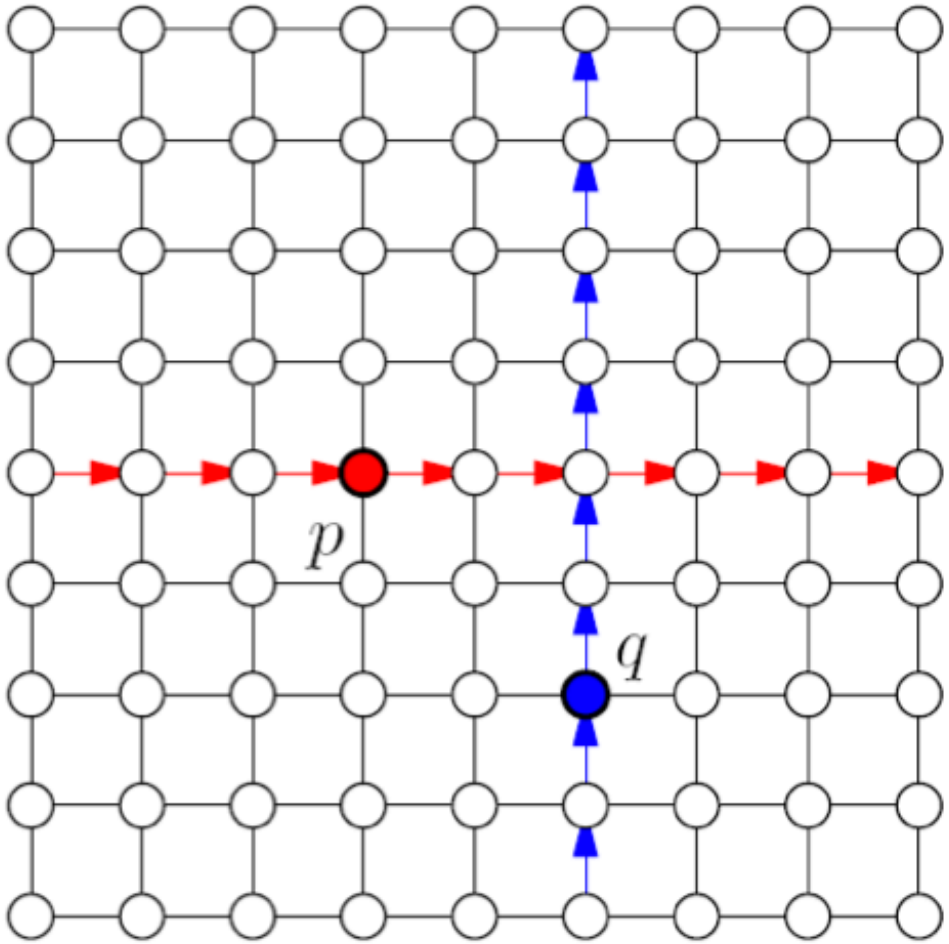
- Each agent carries list of events : can be large
- Random walk can take a long time to reach far away regions
- Harder to analyze for the specific algorithm in the paper
- Inefficient: may visit same nodes many times

# Double rulings

- Store data on a curve, like rumor routing.
- Not a random curve, a more structured approach, like GHT
- The curve depends on the data

# 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|



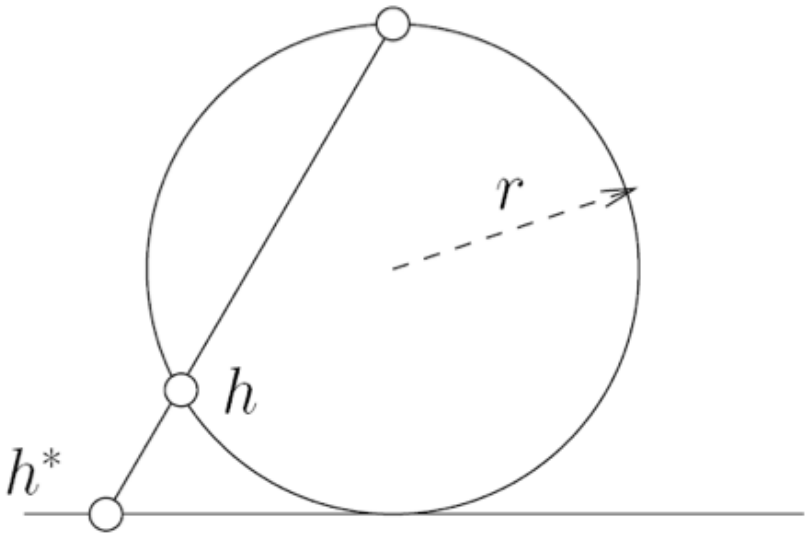
# Spherical Double ruling

- Producer follows a **circle** to a hashed location
  - Includes GHT as subcase
  - Allows a variety of retrieval schemes

- Improves GHT
  - Load balancing for popular data types
  - Distance sensitivity
  - Flexibility: better fault tolerance

# Double rulings on a Sphere

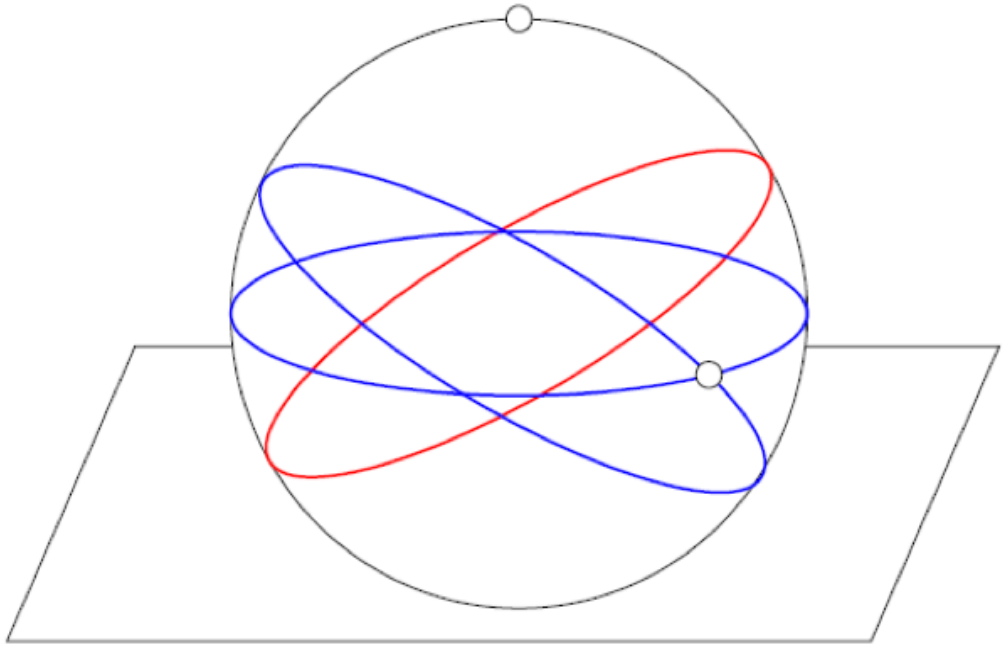
- Stereographic projection maps the plane to sphere
  - Circles map to circle
  - May incur distortion

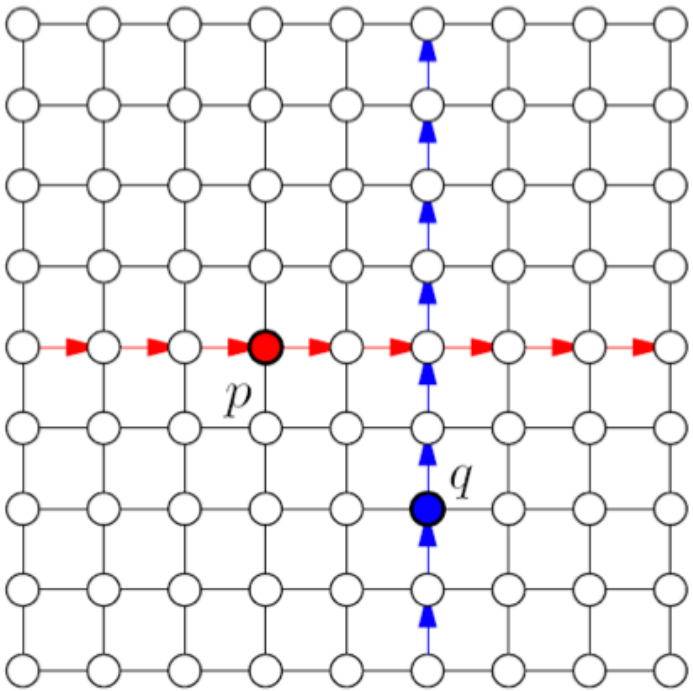



- Network is finite
  - Choose location and size of sphere such that distance distortion is bounded by 1 + ε

# Spherical Double Rulings

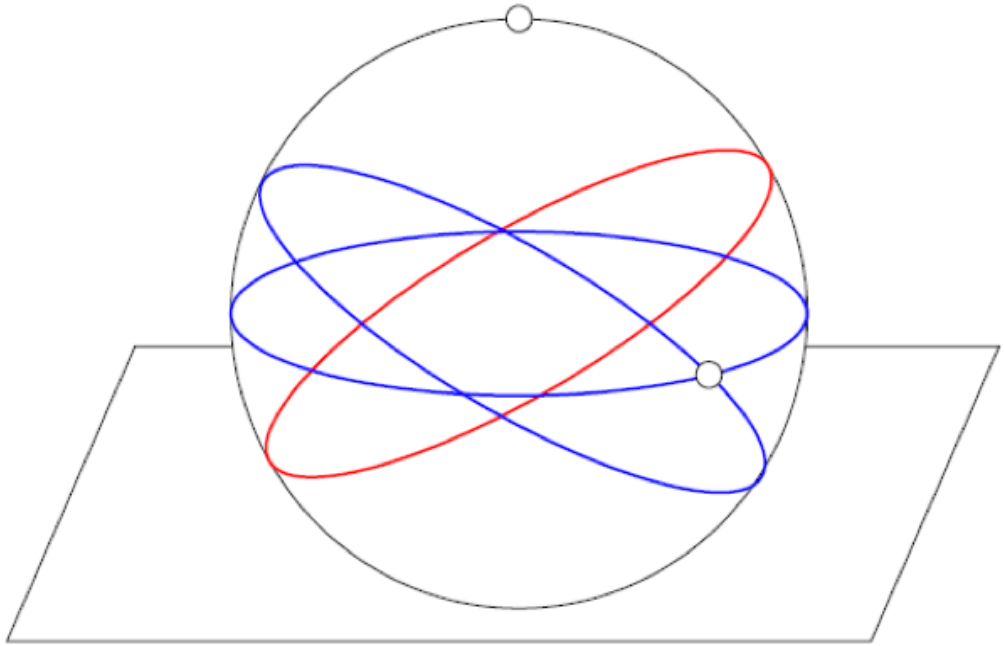
- Any two great circles on a sphere intersect
- Use great circles in place of vertical/horizontal lines

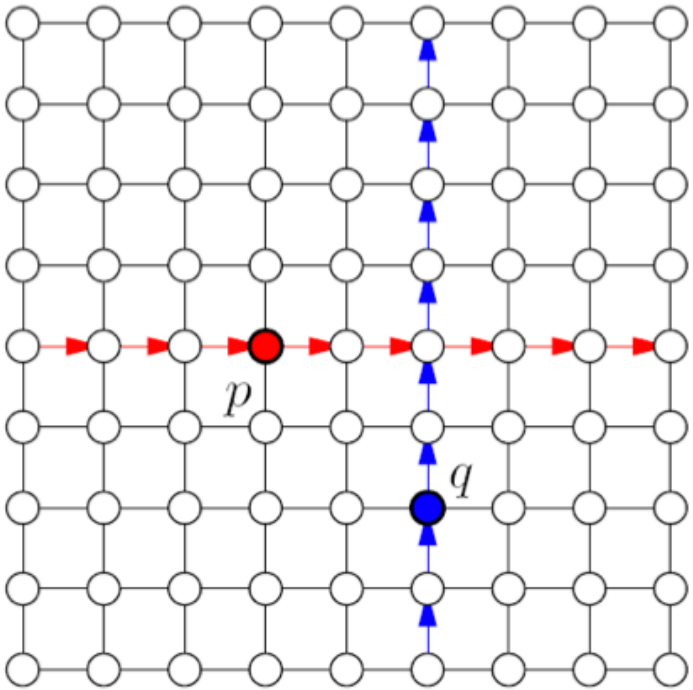




# Spherical double ruling

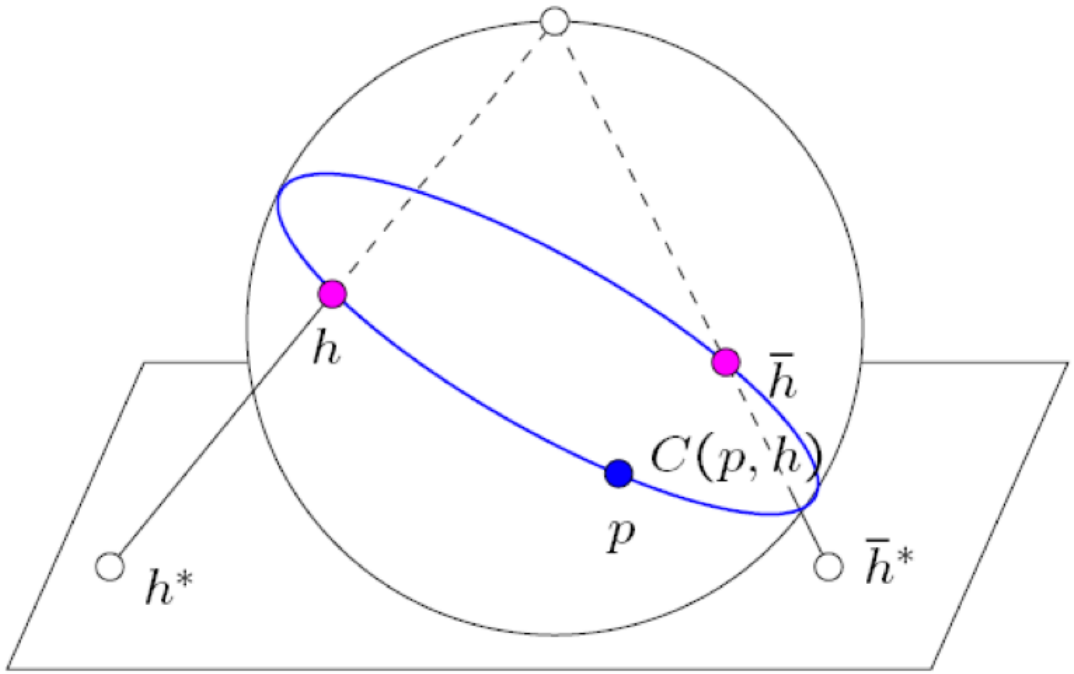
- Difference with Rectilinear double rulings:
  - Infinitely many great circles through a point
  - Much more flexibility





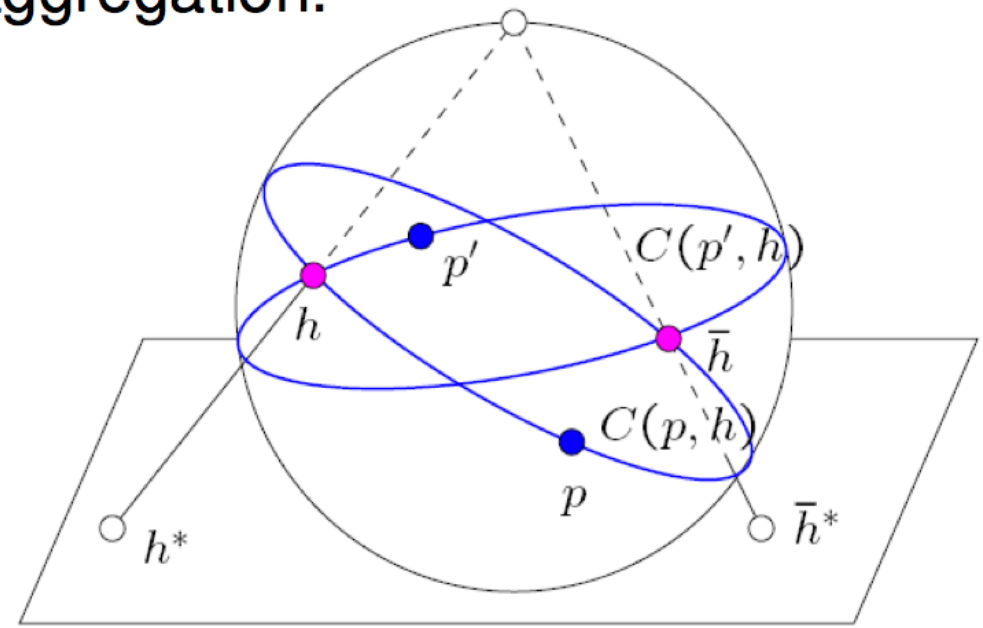
# Data Replication

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

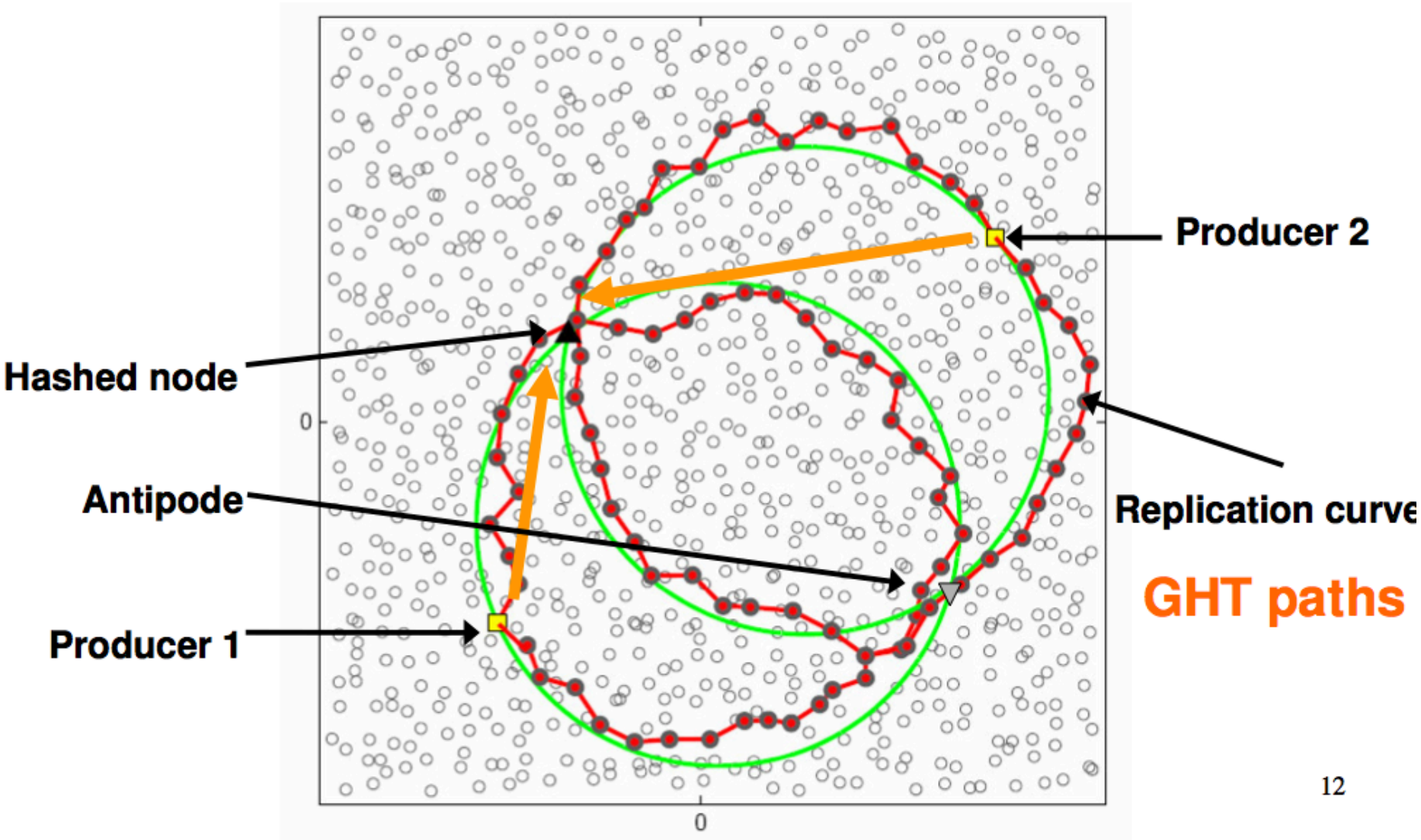


# Data replication

- Different producers with same data type hash to different great circles all passing through hash point $h$ and its antipodal point $\bar{h}$
  - Allow aggregation.



# Examples


Producer 2
Hashed node
Antipode
Replication curve
GHT paths
Producer 1
0
0

# Data Retrieval

- Flexible retrieval rules
  - GHT style retrieval: from a specific point
  - Distance sensitive retrieval
  - Aggregate data retrieval
  - Full power data retrieval

# GHT Style retrieval

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

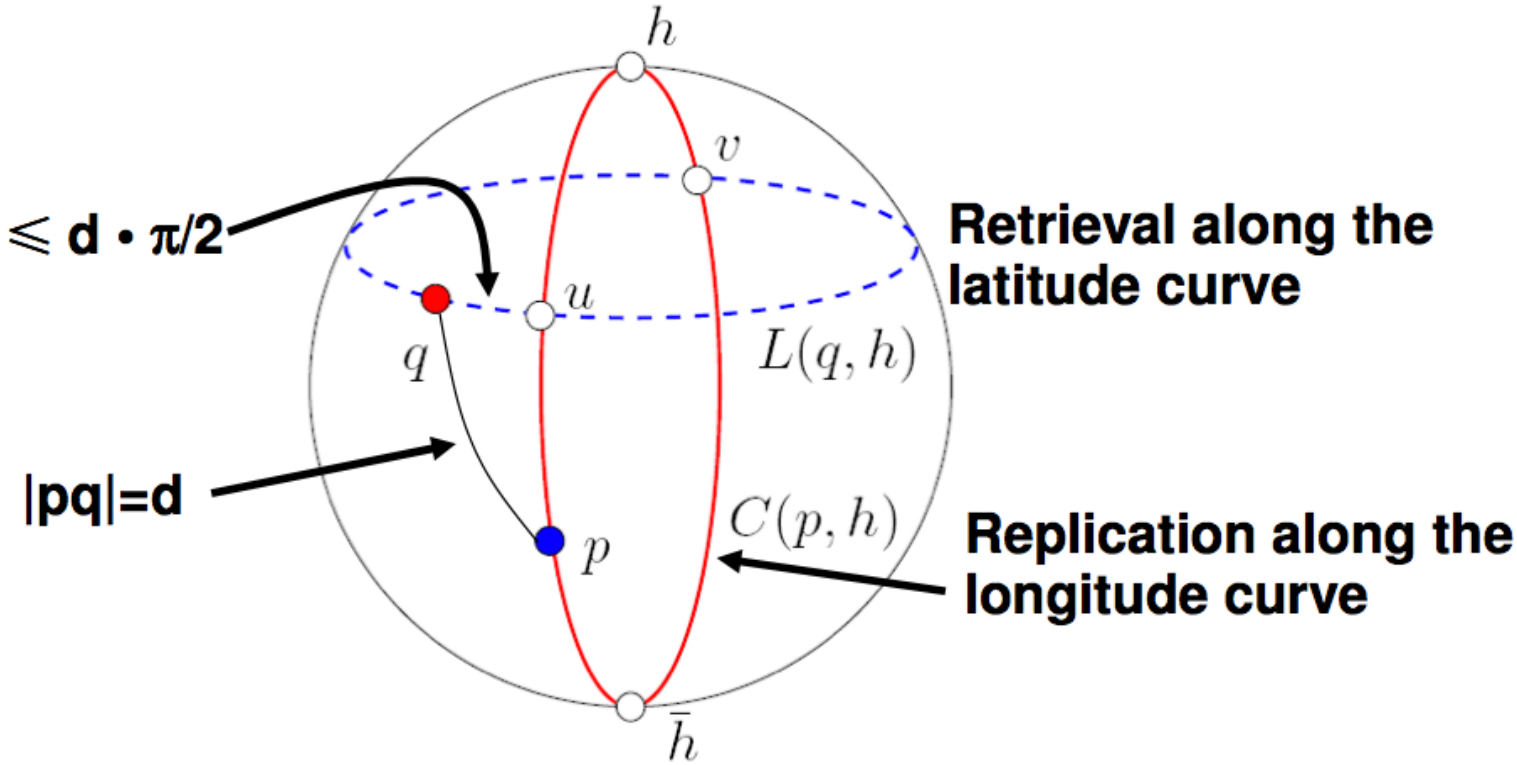
Consumer goes to hashed node $h$ or its antipodal, whichever is closer.

# Distance sensitive retrieval

- Distance Sensitive: If producer is at distance *d* from *q,* consumer should find data at cost *O(d)*
  - Consumes less network resources
  - Consumers are likely to be more interested in immediate vicinity
  - Lower delay : important in quick response

# Distance Sensitive Retrieval

- Rotate the sphere so that hash node is at north pole




**If q is d away from p, the distance from q along latitude curve is $\leqslant$ d • π/2.**

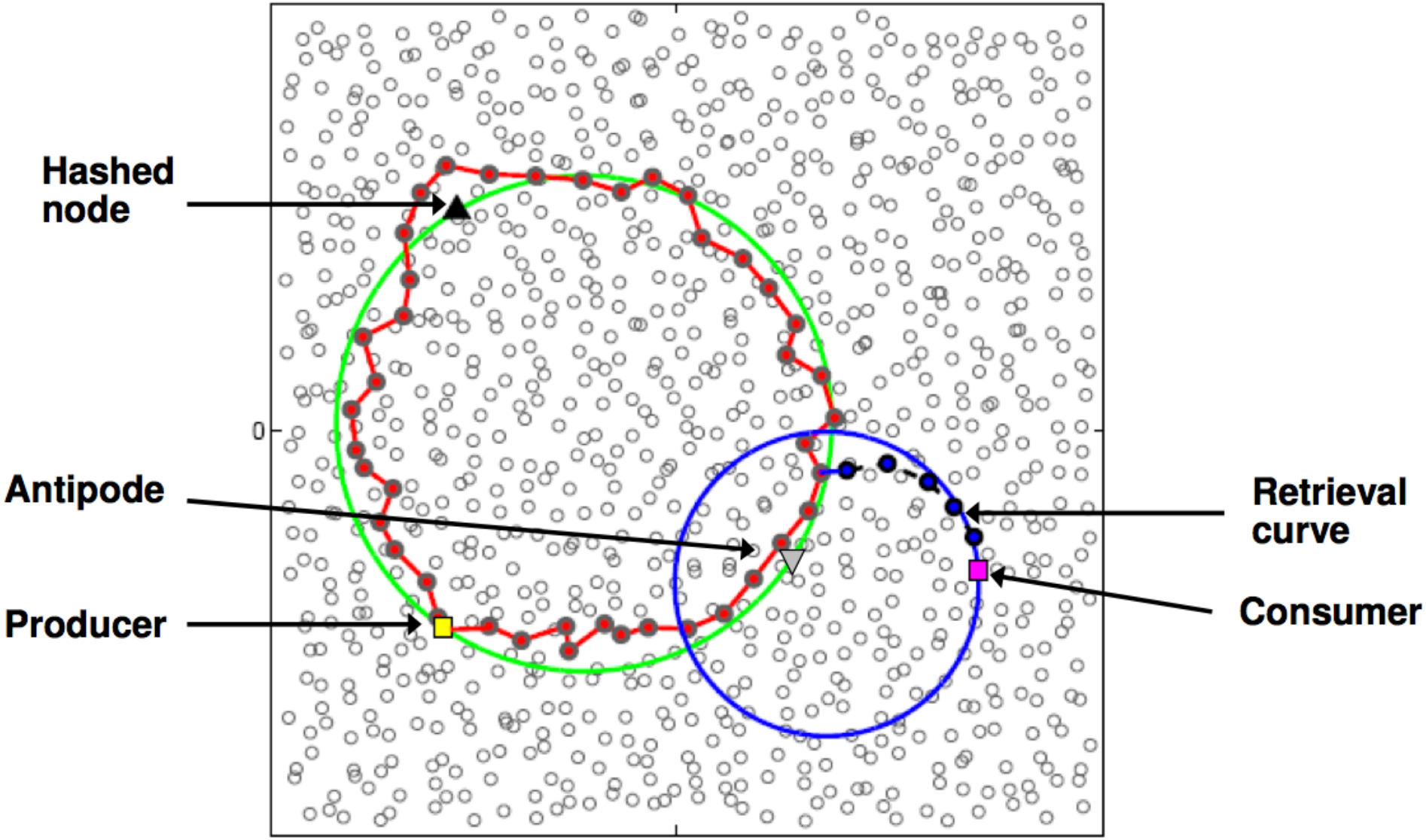
# Distance Sensitive Retrieval

- Distance Sensitive: If producer is at distance *d* from *q,* consumer should find data at 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

# Distance Sensitive Retrieval



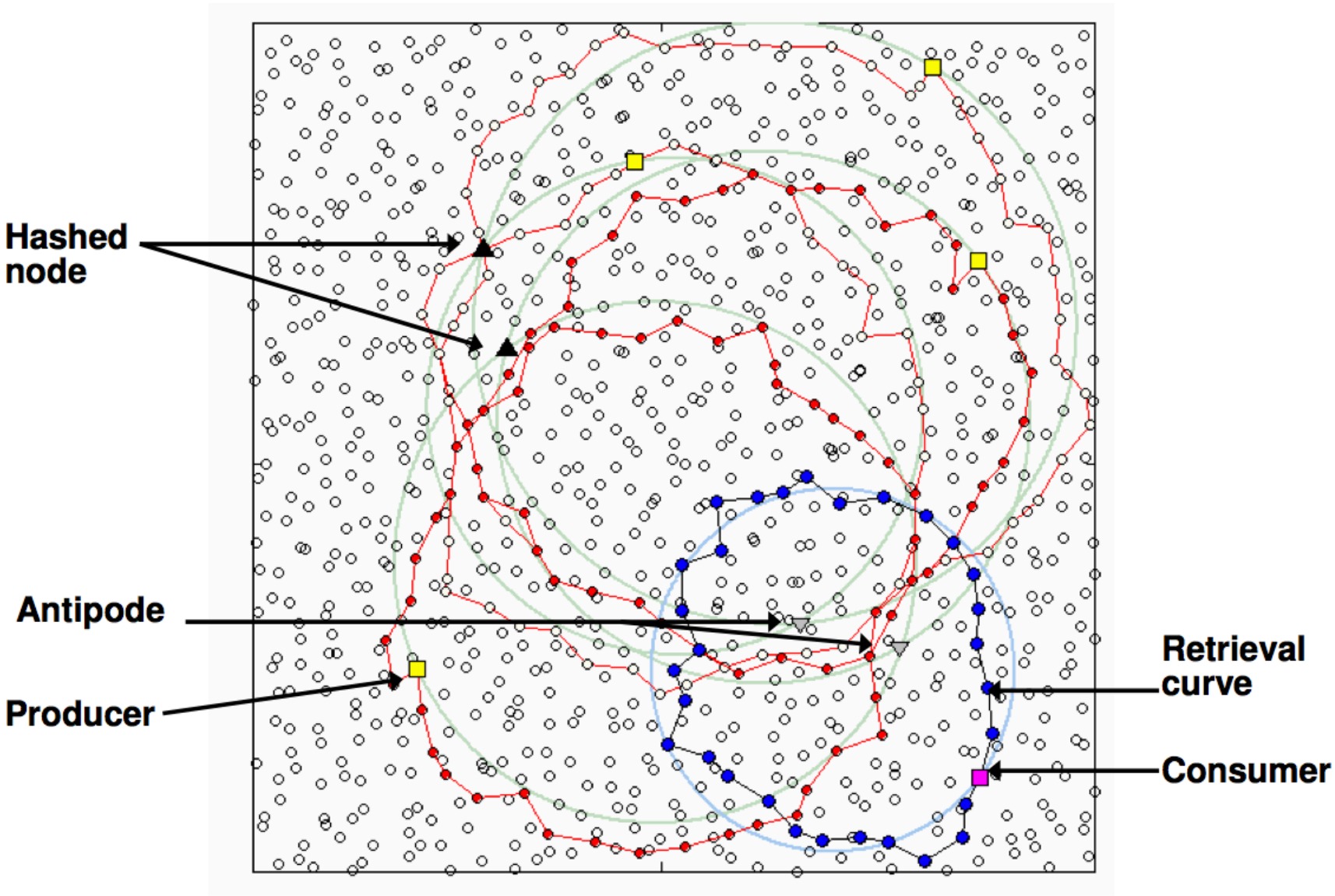
# Aggregate data retrieval

- Consumer wants several types of data
  - E.g. Elephant and lion 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!
- More freedom for consumers and betetr load balancing

# Aggregate data retrieval



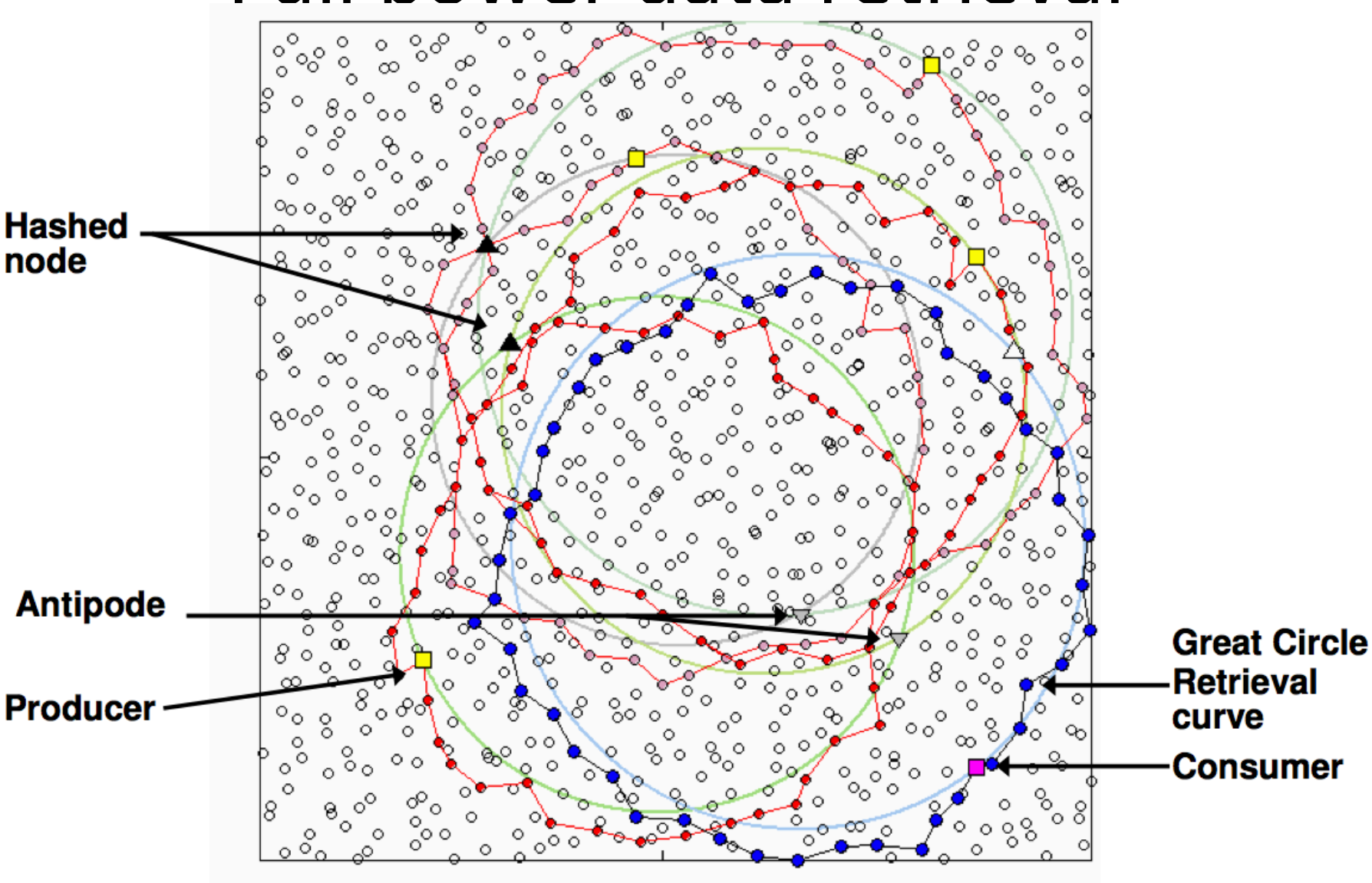
# Full power data retrieval

- Consumer wants all the data

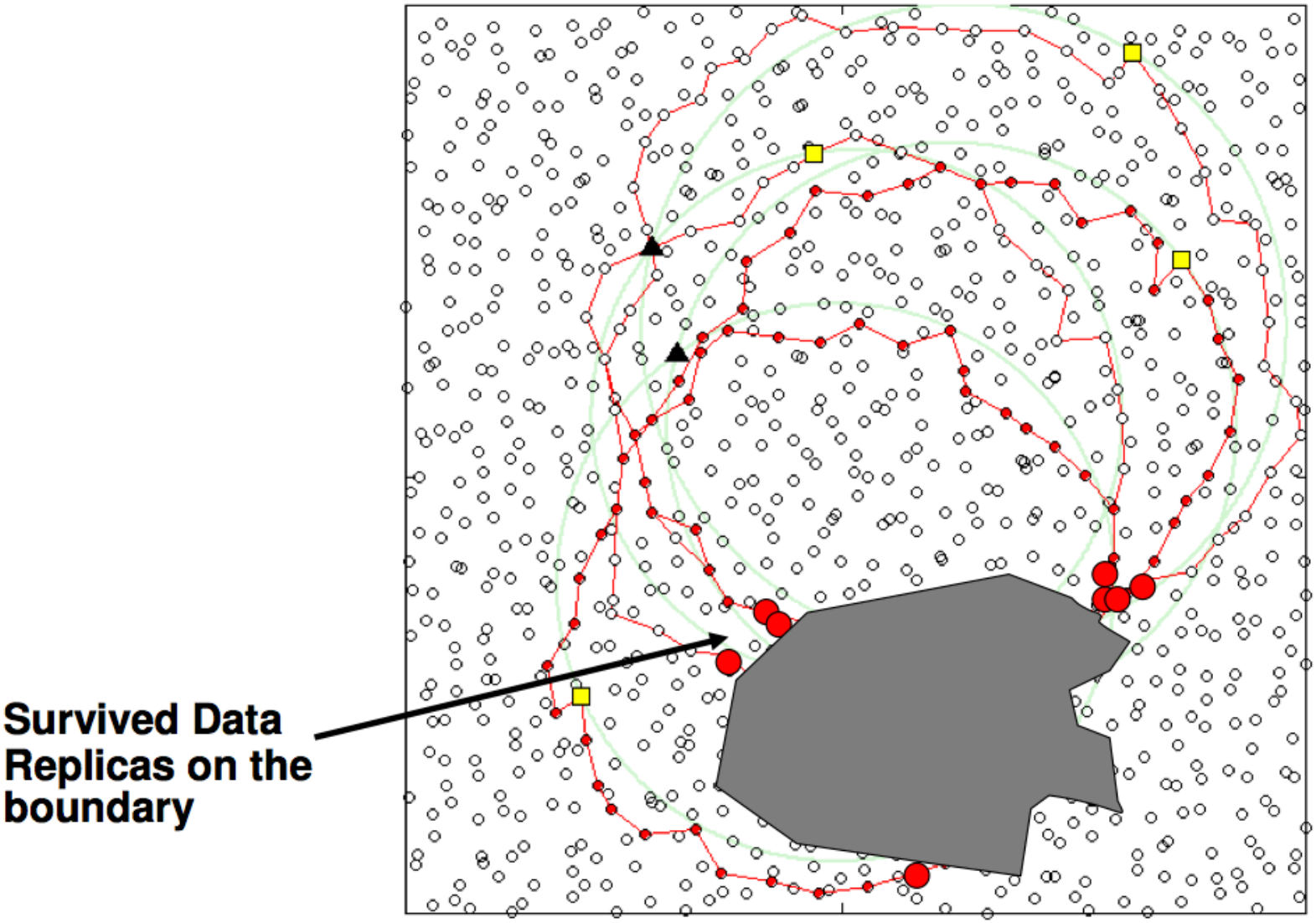
Follow a great circle, retrieve all data.

- Any two great circles intersect
- Many such great circles

# Full power data retrieval

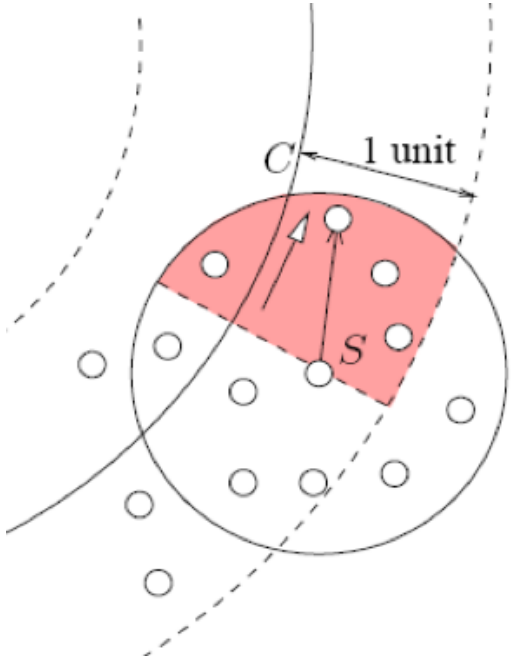


# Local data recovery on node failures



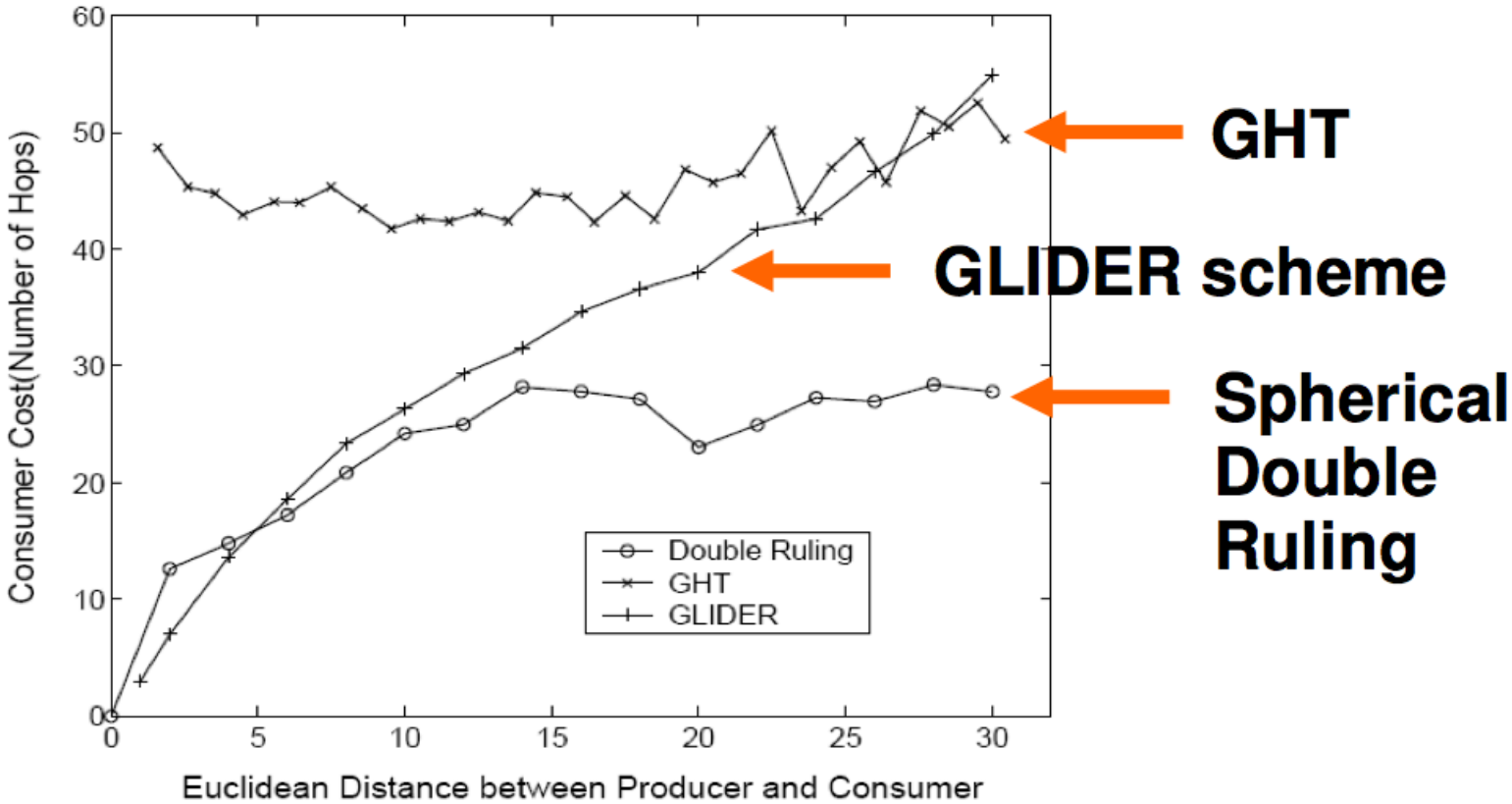
# Implementation

- How to forward data on a virtual curve?
- Use geographic greedy forwarding on a curve



# Simulations: Distance Sensitivity

**4200 nodes with average degree 8 per node.**

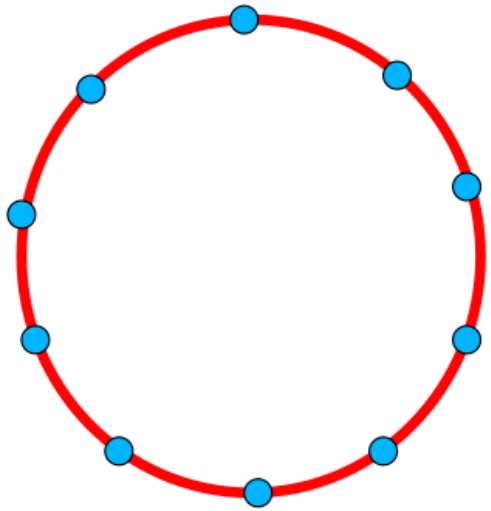



**Distance Sensitivity of queries**

GLIDER based scheme : Q. Fang et. al. Landmark-based information storage and retrieval in sensor networks. *INFOCOM* 2006.

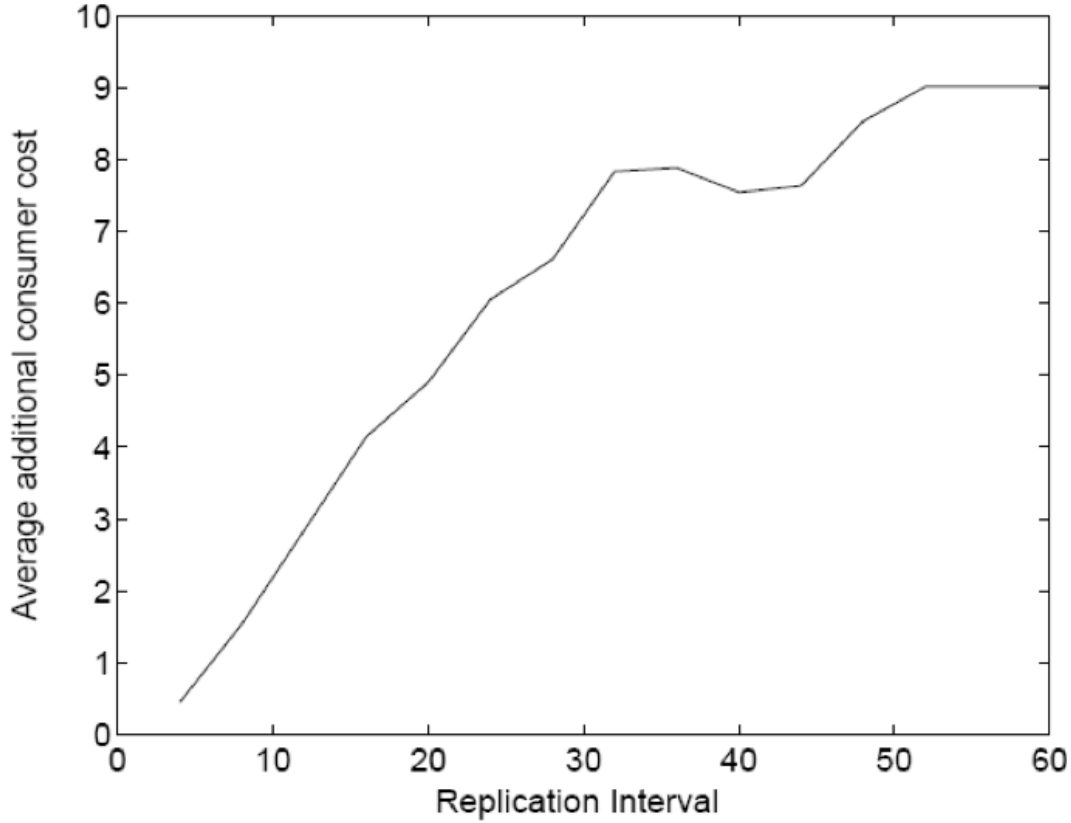
# Simulation: Storage retrieval tradeoff

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



**Replication Interval**

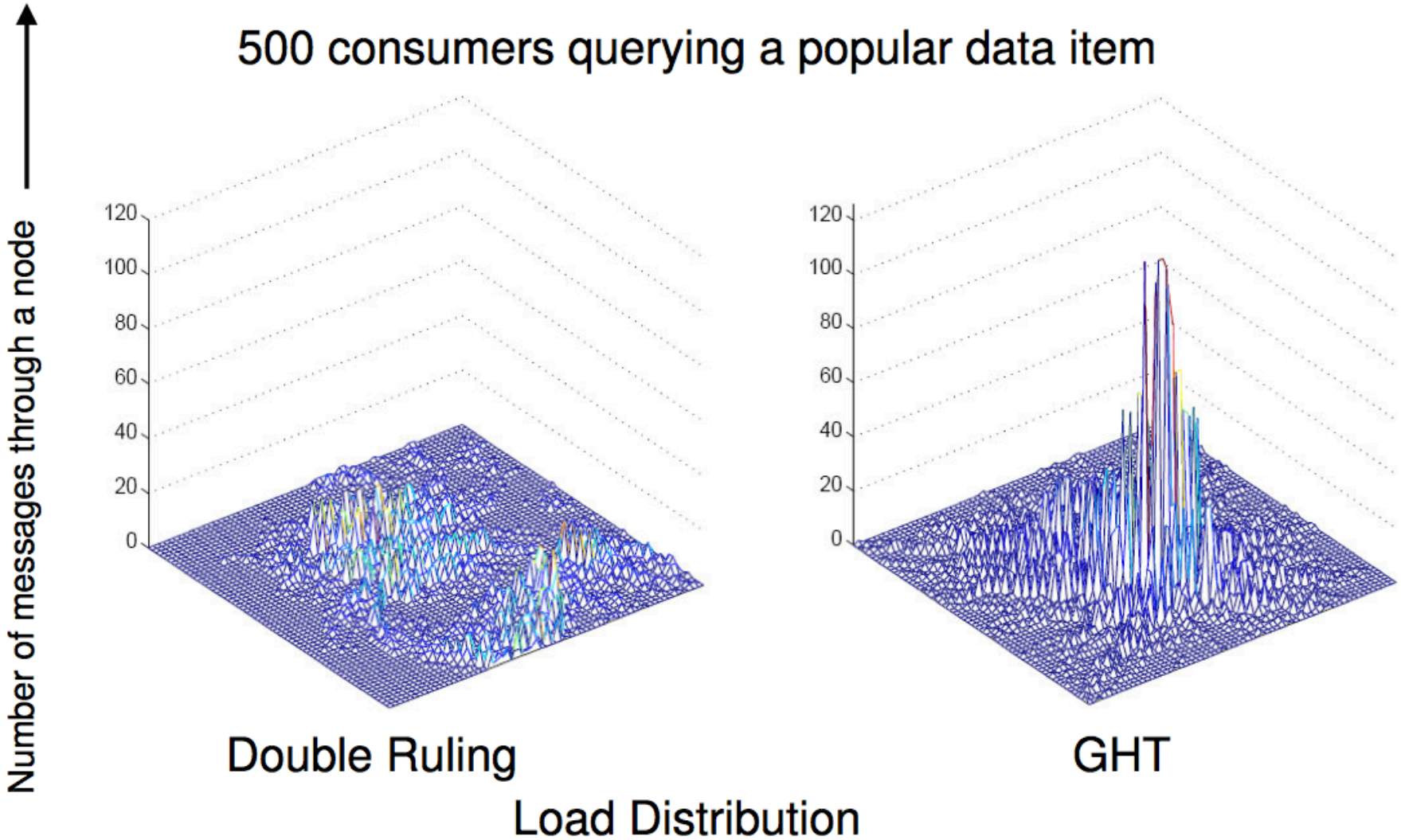
Increasing Consumer cost




Larger Replication Interval

Decreasing Storage Cost

# Simulation: Load balancing



# Discussion

- Data collection by mobile data mules
  - Physically move along any retrieval curve
- Advanced hashing
  - Make sure that similar data are stored nearby
- Networks with holes
  - Require better strategies

# Project ideas handout

- Make a group of 1, 2 or3 people
- Decide on a project
- Email your choice
- Do some reading (search on web)
- Send a short (1/2 page to 1 page) proposal by December 1