DONUT: Efficient routing for decentralized range queries

Sergey Legtchenko, Sébastien Monnet, Pierre Sens

LIP6/INRIA

Distributed Range Querying

- Context : Large Scale decentralized systems (overlays) storing data.
- Goal : Retrieve a set of objects satisfying a given condition
  e.g., Get(o, where o.x $\in$ [1..10]*)*

Range Querying is an *extremely* widespread applicative requirement.

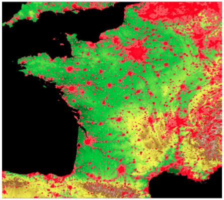
Range query support : a widespread requirement

- **Multiplayer Online Games.** retrieve all the objects located in a virtual room : Get(objects o where $o.x \in [1..10]$ and $o.y \in [3..5]$ and $o.z \in [100..140]$)
- **File Sharing.** efficient file search : Get(files f where f.type=song and f.year=2011)
- **Distributed file systems.** Get(files f where f.type=txt and f.directory=/ and f.owner=root)
- **Distributed Databases.** select account where balance<0 and lastModification>1day ;

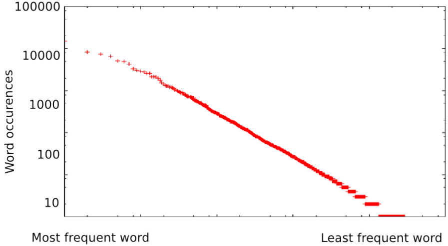
Main challenge of range-query overlays : non uniform data distributions

- Semantically close objects should be close to each other in the overlay.
- Object placement depends on its semantic attributes.
- Consequence : non uniform object distributions.

French population



Word occurences in "Ulysses" by James Joyce

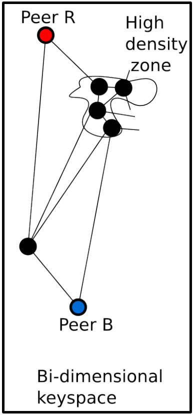


Second Life avatars

Consequence of non uniform keyspace density

Simple Scenario :

- The blue peer B needs to send a message to the red peer R.
- The peers use **greedy routing**.
  i.e., at each hop, the *semantic* distance to destination is minimized.

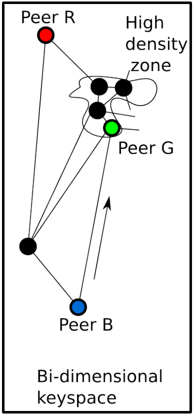


Consequence of non uniform keyspace density

Simple Scenario :

- The blue peer B needs to send a message to the red peer R.
- The peers use **greedy routing**.
  i.e., at each hop, the *semantic* distance to destination is minimized.

B's neighbor G is *semantically* the closest to R.

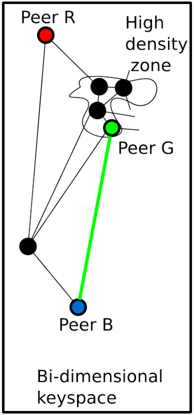


Consequence of non uniform keyspace density

Simple Scenario :

- The blue peer B needs to send a message to the red peer R.
- The peers use **greedy routing**.
  i.e., at each hop, the *semantic* distance to destination is minimized.

B's neighbor G is *semantically* the closest to R.

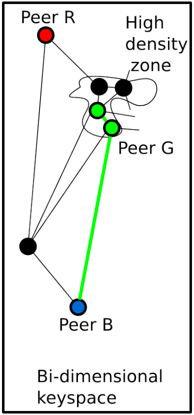


Consequence of non uniform keyspace density

Simple Scenario :

- The blue peer B needs to send a message to the red peer R.
- The peers use **greedy routing**.
  i.e., at each hop, the *semantic* distance to destination is minimized.

B's neighbor G is *semantically* the closest to R.

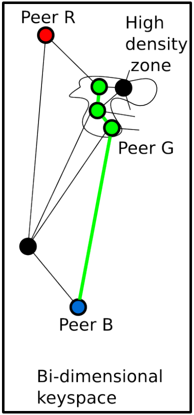


Consequence of non uniform keyspace density

Simple Scenario :

- The blue peer B needs to send a message to the red peer R.
- The peers use **greedy routing**.
  i.e., at each hop, the *semantic* distance to destination is minimized.

B's neighbor G is *semantically* the closest to R.



Consequence of non uniform keyspace density

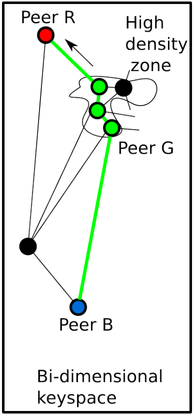
Simple Scenario :

- The blue peer B needs to send a message to the red peer R.
- The peers use **greedy routing**.
  i.e., at each hop, the *semantic* distance to destination is minimized.

B's neighbor G is *semantically* the closest to R.

The routing process is not efficient.

- The route length is 4 hops.
- The optimal path length is 2 hops.



Problem analysis

Intuition :

The peers are not “aware” of density zones while routing messages.

Problem analysis

Intuition :

The peers are not "aware" of density zones while routing messages.

More precisely :

- Greedy routing minimizes the *semantic* distance.
- But in heterogeneous data distributions, semantic distance is **not** proportional to hop distance.

Problem analysis

Intuition :

The peers are not “aware” of density zones while routing messages.

More precisely :

- Greedy routing minimizes the *semantic* distance.
- But in heterogeneous data distributions, semantic distance is **not** proportional to hop distance.

Consequence :

Minimizing semantic distance does not necessarily minimize hop distance.

Plan

Outline of the solution

Idea : Make routing "density aware" by building a density map on each peer.

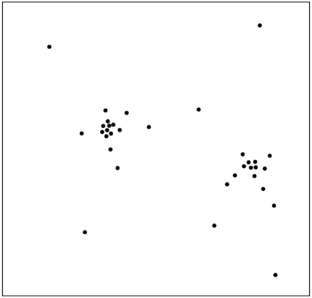
On each peer :

1. Build a map of the density distribution.
2. Estimate hop distance using the map.
3. Build efficient density aware shortcuts in the overlay graph.

Making peers "density aware"

Naive solution : Store coordinates of *all* peers

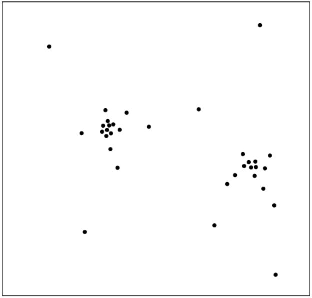
Heterogeneous peer distribution



Making peers "density aware"

Naive solution : Store coordinates of *all* peers

Heterogeneous peer distribution



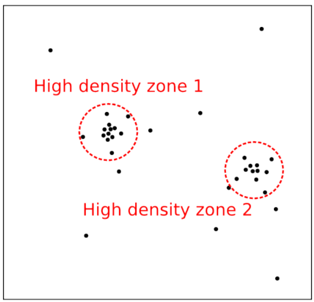
Problem : not scalable !

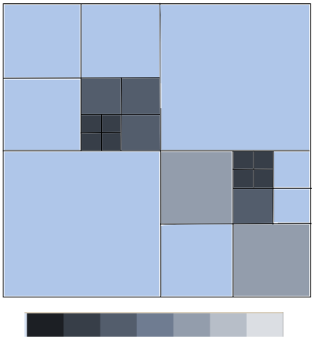
- Overlay size : possibly many thousands of peers.
- High churn rates cause frequent coordinate updates.

Need to approximate the distribution.

Making peers "density aware"

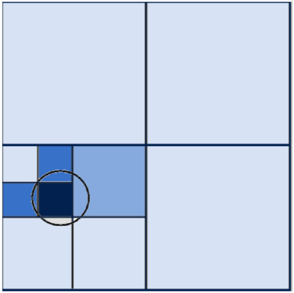
Our solution : Approximate the density distribution.





Bootstrap of the density map

- At bootstrap, the density map is empty.
- The peer computes the surrounding density using the coordinates of its direct neighbors.
- The peer adds the density information to the map.

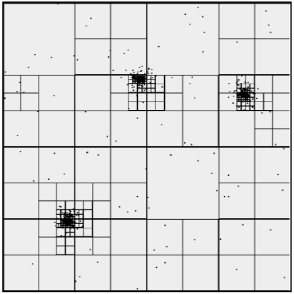


The density map at bootstrap on the peer P. The circle represents the density of the region known by P.

Propagating density information

The peers share their density information to build the full map.

- By adding density information to lookup messages
- By using a gossip algorithm that propagates the most recently modified map-regions.



Snapshot of the density-map taken from a peer during our simulation.

Handle keyspace evolution

The keyspace density may change :

- Objects can move in game.
- Objects can be deleted.
- New objects can be uploaded to the overlay.

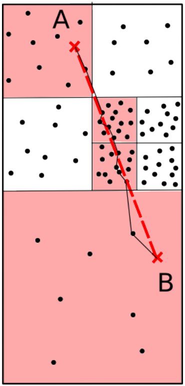
The density map is able to evolve :

- Each density zone of each map is timestamped.
- If a peer receives a more recent information about a region, the old information is erased from the map.

Estimate hop distance using the density map

Three-step **local** algorithm to estimate distance between two coordinates A & B :

1. Find the regions that intersect [AB].
2. Estimate the number of hops in each region *given its density and size.*
3. Sum the hops of all regions to obtain an estimation of the global number of hops.

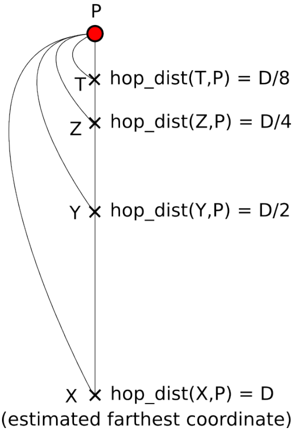


Create density-aware shortcuts in the overlay

The shortcut algorithm *locally* finds :

- an estimation of the coordinate X that is the farthest in terms of overlay distance.
- peers that are estimated to be at $\frac{D}{2^p}$ hops from P, with $p \in [1..log(N)]$ (N : number of peers).

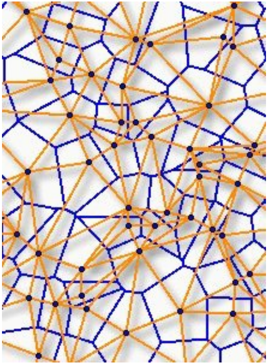
If the estimation is accurate enough, the formed shortcuts follow the 2-Harmonic distribution, known to enable decentralized **(poly)logarithmic routing**.



Plan

Evaluation Configuration

- Keyspace : Bi-dimensional (map structure : QuadTree).
- Overlay : based on a Delaunay triangulation.
- Testbed : PeerSim, a widespread event-based simulator.



Trace Injection

Latency traces :

- collected between 2500 hosts spread over the internet.

Churn traces :

- Overnet, Skype, Microsoft corporate desktops

Keyspace density traces :

- Traces derived from Second Life avatar distribution.

Shortcut efficiency is compared to state-of-the-art techniques

Random shortcut distribution :

- Shortcut coordinates chosen uniformly at random in the keyspace.

Shortcut efficiency is compared to state-of-the-art techniques

Random shortcut distribution :

- Shortcut coordinates chosen uniformly at random in the keyspace.

Techniques to reach a 2-Harmonic distribution :

- Uniform : The hop-distance is considered to be proportional to distance between coordinates.

Shortcut efficiency is compared to state-of-the-art techniques

Random shortcut distribution :

- Shortcut coordinates chosen uniformly at random in the keyspace.

Techniques to reach a 2-Harmonic distribution :

- Uniform : The hop-distance is considered to be proportional to distance between coordinates.
- Near optimal : The exact hop-distance to each peer is known (not scalable in practice).

Shortcut efficiency is compared to state-of-the-art techniques

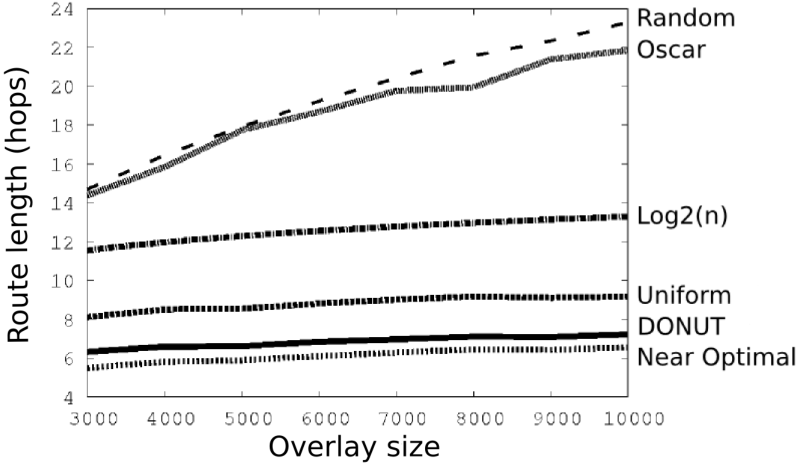
Random shortcut distribution :

- Shortcut coordinates chosen uniformly at random in the keyspace.

Techniques to reach a 2-Harmonic distribution :

- Uniform : The hop-distance is considered to be proportional to distance between coordinates.
- Near optimal : The exact hop-distance to each peer is known (not scalable in practice).
- Oscar : The keyspace is sampled by random walks while building the shortcuts, *but the information is lost after rewiring*.

Average route length, in hops vs overlay size

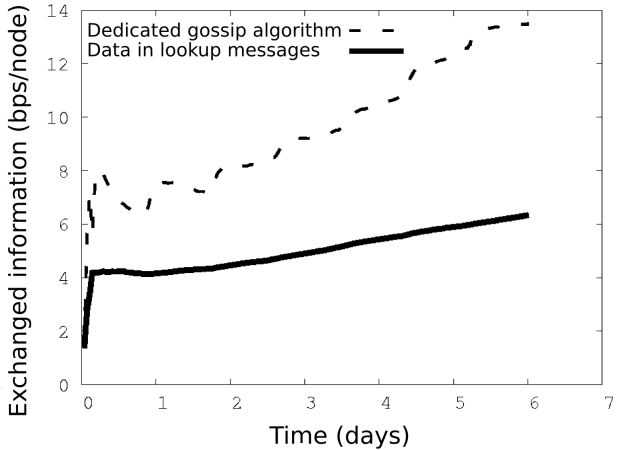


Average route latency in milliseconds. Overlay size : 2500 peers

Strategy	Average Latency	Standard Deviation
Random	464.5	41.1
Oscar	438.2	44.6
Uniform	370.2	31.8
DONUT	300.7	27.3
Near Optimal	295.6	18.9

FIGURE: Overnet churn trace and real latency matrix

Data exchanged for density-map propagation, 2500 peers



Conclusion

A novel technique to retrieve and store multi-dimensional keyspace density-information.

- Cheap : a few bytes per second per peer.
- Accurate : Helps decreasing latency by nearly 20% compared to state-of-the-art.

The map can be used to gather other distributed information :

- Infromation about peer load to enable efficient reactive load balancing.
- Object popularity distribution, etc.

Thank you for your attention !

1. Defining the problem
2. Our solution
3. Evaluation

Questions.

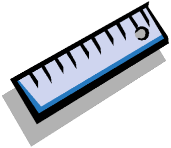
Evaluation Metrics

Density map propagation :

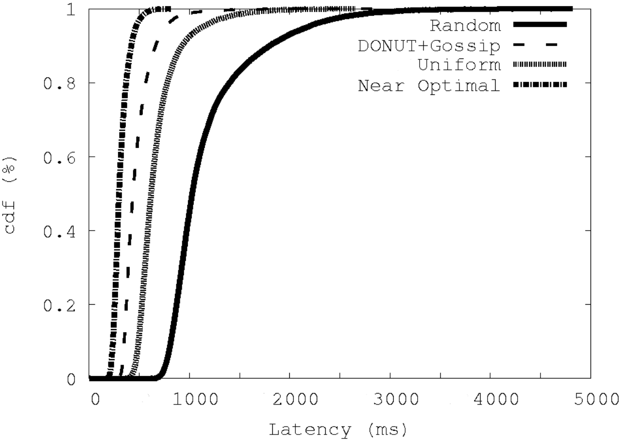
- Amount of exchanged data
- Route latency distribution

Routing efficiency :

- Average route length
- Average route latency



Average latency distribution



Range-Query overlay architecture overview

Similar to Distributed Hash Tables (DHTs) :

- Objects and peers have coordinates in the same metric space (called *keyspace*).
- Each peer is responsible for a space-range centered on its coordinates and stores all the objects in the range.

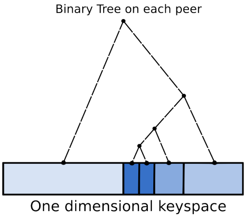
Key-Based Routing

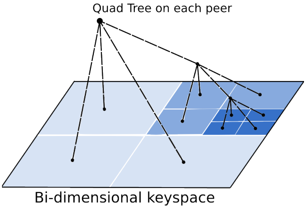
- All queries have keyspace-coordinates
- A peers forwards a query to its neighbor which coordinate is the closest to the query's coordinate.

A tree to build the density map

- The nodes have 0 or $2 \times dim_{keyspace}$ children.
- Each node is responsible for a region of the keyspace.
- Leaves store the average density measured in their region.
- The map precision increases with the tree depth.

Possibility to have a *lightweight approximation* of the keyspace density.





Data distribution model

Unlike DHTs :

- Object's coordinates are related to the semantic properties of the object.
- Semantically close objects are stored on contiguous nodes.
- N object semantic attributes $\Rightarrow$ N keyspace dimensions.

Unlike in DHTs, uniform density of the keyspace is **not** ensured.