Landmark-based routing
Landmark-based routing


[Fonseca05] Rodrigo Fonseca, Sylvia Ratnasamy, Jerry Zhao, Cheng Tien Ee and David Culler, Scott Shenker, Ion Stoica, Beacon Vector Routing: Scalable Point-to-Point Routing in Wireless Sensornets, NSDI'05. Landmark-based routing scheme.

Landmark-based schemes

- k nodes are selected as landmarks (beacons) that flood the network. Each node records hop distances to these landmarks.
  - estimate pair-wise distances,
  - point-to-point routing.

- Pros:
  - simplicity,
  - location-free,
  - independent of dimensionality (works for 3D networks).
  - No unit disk graph assumption
Use landmarks to estimate pair-wise distances

- Triangulation: estimate via triangle inequality
  - \((u,v), \text{ beacon } b : |d(u,b) - d(v,b)| \leq d(u,v) \leq d(u,b) + d(v,b)\)
  - lower bound: \(d^-(u,v) = \max_{\text{beacons } b} |d(u,b) - d(v,b)|\)
  - upper bound: \(d^+(u,v) = \min_{\text{beacons } b} d(u,b) + d(v,b)\)
  - Internet setting, IDMaps [Francis+ ’01], etc

- magic: relative error <1 on 90% node pairs
  - 900 random nodes, 15 beacons
  - relative error\((x,y) = |x-y| / \min(x,y)\)
A simple case

- With O(1) random landmarks, $d^+(u,v) \leq 3d(u,v)$ for all but $\varepsilon$ fraction of pairs with prob $1-\gamma$.
  - At least one beacon inside $B(u)$.
  - For any point $v$ outside $B(u)$,
    - $d(v,b) \leq d(u,b) + d(u,v) \leq 2d(u,v)$
    - $d^+(u,v) \leq d(u,b) + d(v,b) \leq 3d(u,v)$

$B(u)$: ball with $\varepsilon n$ nodes inside.
Sensor networks, doubling metric

- A metric space has doubling dimension $s$ if any ball of radius $r$ can be covered by $s$ balls of radius $r/2$.
  - Geometric growth.
  - Binary trees do not have constant doubling dimension.
  - Euclidean space has constant doubling dimension.
  - Many practical networks fit in this model: Internet delay distance, sensor network (not too fragmented), VSLI layout, etc.
Improved bound on triangulation

- With $O(1)$ random landmarks, $d^+(u,v) \leq (1+\delta) d^+(u,v)$ for all but $\epsilon$ fraction of pairs with prob $1-\gamma$. 
Landmark-based embedding

Global Network Positioning (GNP) [Ng+Zhang’02]

• select small #nodes as “beacons”
  – users measure latencies to beacons
• embed into low-dim Euclidian space
  – embed the beacons
  – embed non-beacons one by one

• magic: 90% node pairs are embedded with relative error < .5
  – 900 random nodes, 15 beacons, 7 dimensions
  – Proof omitted.
Discussion

• With distances estimation, potentially we can do “greedy routing”.

• Next: different ways to implement landmark-based routing schemes.
  – Beacon Vector Routing
  – GLIDER
  – S4: compact routing
Beacon Vector Routing (BVR)

- A heuristic landmark-based routing.
- Every node remembers hop counts to a total of $r$ landmarks.
  - Move towards a beacon when the destination is closer to the beacon than the current node
  - Move away from a beacon when the destination is further from the beacon than the current node
Beacon Vector Routing (BVR)

• A heuristic landmark-based routing.
• Every node remembers hop counts to a total of \( r \) landmarks.
• Routing metric:
  – Pulling landmarks (those closer to destination).
    \[
    \delta^+(p, d) = \sum_{i \in C_k(d)} \max(p_i - d_i, 0)
    \]
    Dist from \( p \) to landmark \( i \).
    Dist from \( d \) to landmark \( i \).
  – Pushing landmarks (further to destination)
    \[
    \delta^-(p, d) = \sum_{i \in C_k(d)} \max(d_i - p_i, 0),
    \]
Beacon Vector Routing (BVR)

- **Routing metric:** Choose a neighbor that minimizes

\[
\delta_k = A\delta^+_k + \delta^-_k
\]

\[
\delta^+_k(p, d) = \sum_{i \in C_k(d)} \max(p_i - d_i, 0)
\]

\[
\delta^-_k(p, d) = \sum_{i \in C_k(d)} \max(d_i - p_i, 0),
\]

- **No theoretical understanding of the performance.**
Fall back mode

• If greedy routing with the routing metric gets stuck, then send to the closest beacon.

• The closest beacon does a scoped flooding.
Simulations

• Assumptions for high level simulation
  – Fixed circular radio range
  – Ignore the network capacity and congestion
  – Ignore packet losses
• Place nodes uniformly at random in a square planner region
  – 3200 nodes uniformly distributed in a 200 * 200 unit area
  – Radio range is 8 units
  – Average node degree is 16
• Vary #total beacons and #routing beacons
Greedy success rate

The graph shows the success rate of Greedy flooding with varying numbers of routing beacons. The y-axis represents the success rate without flooding, and the x-axis represents the number of beacons. The graph includes lines for 5, 10, and 20 routing beacons, with a legend indicating each line type for these categories. The graph also includes a line for true positions, which is flat above 90%.
Greedy routing with 10 beacons

The graph illustrates the success rate without flooding against the number of beacons for different density conditions. The lines represent:
- True positions, high density
- True positions, low density
- High Density (15.7)
- Low Density (9.8)
Obstacles

<table>
<thead>
<tr>
<th>Length of Obstacles</th>
<th>Number of Obstacles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0.96 (0.98)</td>
</tr>
<tr>
<td>20</td>
<td>0.96 (0.98)</td>
</tr>
</tbody>
</table>

Table 4: Comparing BVR with greedy forwarding over true positions in the presence of obstacles
Routing around holes

• Real-world deployment is not uniform, has holes (due to buildings, landscape variation).
• Face routing is too “Short-sighted” and greedy.
• Boundary nodes get overloaded.
GLIDER

• 2-level infrastructure

• Top-level: capture the global topology.
  – Where the holes are (e.g., CS building, Javitz center, etc).
  – General routing guidance (e.g., get around the Javitz center, go straight to SAC).

• Bottom-level: capture the local connectivity.
  – Gradient descent to realize the routing path.
2-level infrastructure

• Why this makes sense?
  – Global topology is **stable** (the position of buildings are unlikely to change often).
  – Global topology is **compact** (a small number of buildings)

• From each node’s point of view:
  – A rough guidance.
  – Local greedy rule.
Combinatorial Delaunay graph

- Given a communication graph on sensor nodes, with path length in shortest path hop counts
- Select a set of landmarks
- Landmarks flood the network. Each node learns the hop count to each landmark.
- Construct Landmark Voronoi Complex (LVC)
Combinatorial Delaunay graph

- Construct **Landmark Voronoi Complex (LVC)**

- Each sensor identifies its closest landmark.

- A sensor is on the boundary if it has 2 closest landmarks.

- If flooding are synchronized, then restricted flooding up to the boundary nodes is enough.
Combinatorial Delaunay graph

- Construct **Combinatorial Delaunay Triangulation (CDT)** on landmarks

- If there is at least one boundary node between landmark $i$ and $j$, then there is an edge $ij$ in CDT.

- Holes in the sensor field map to holes in CDT.

- CDT is broadcast to the whole network.
Virtual coordinates

Each node stores virtual coordinates \((d_1, d_2, d_3, \ldots d_k)\),

\[d_k = \text{hop count to the } k\text{th reference landmark (home+neighboring landmarks)}\]

home landmark
(think about post-office)

resident tile

reference landmarks

Boundary nodes
Theorem: If $G$ is connected, then the Combinatorial Delaunay graph $D(L)$ for any subset of landmarks is also connected.

1. Compact and stable
2. Abstract the connectivity graph:
   Each edge can be mapped to a path that uses only the nodes in the two corresponding Voronoi tiles; Each path in $G$ can be “lifted” to a path in $D(L)$
Information Stored at Each Node

- The shortest path tree on $D(L)$ rooted at its home landmark
- Its coordinates and those of its neighbors for greedy routing
Virtual coordinates

• With the virtual coordinates, a node can test if
  – It is on the boundary (two closest landmarks).
  – A neighbor who is closer to a reference landmark.
Local Routing with Global Guidance

- Global Guidance: routing on tiles
  the D(L) that encodes global connectivity information is accessible to every node for proactive route planning on tiles.

- Local Routing: how to go from tile to tile.
  high-level routes on tiles are realized as actual paths in the network by using reactive protocols.
GLIDER -- Routing

1. Global planning

2. Local routing
   - Inter-tile routing
   - Intra-tile routing
Intra-tile routing

• How to route from one node to the other inside a tile?

• Each node knows the hop count to home landmark and neighboring landmarks.

• No idea where the landmarks are.

• A bogus proposal: p routes to the home landmark then routes to q.
Centered Landmark-Distance Coordinates and Greedy Routing

Reference landmarks: $L_0, \ldots L_k$

$T(p) = L_0$

Let $s = \text{mean}(pL_0^2, \ldots, pL_k^2)$

**Local virtual coordinates:**

$c(p) = (pL_0^2 - s, \ldots, pL_k^2 - s)$

(centered metric)

**Distance function:**

$d(p, q) = |c(p) - c(q)|^2$

**Greedy strategy:** to reach $q$, do gradient descent on the function $d(p, q)$
Local Landmark Coordinates – No Local Minimum

- **Theorem**: In the continuous Euclidean plane, gradient descent on the function $d(p, q)$ always **converges to the destination** $q$, for at least three non-collinear landmarks.

- **Landmark-distance coordinates**

  $$[B(p)]_i = |p|^2 - 2p \cdot u_i + |u_i|^2$$

- **Centered coordinates**

  $$[C(p)]_i = -2p \cdot (u_i - \bar{u}) + w_i$$

  $$\bar{u} = \frac{1}{k} \sum_j u_j \quad w_i = |u_i|^2 - \frac{1}{k} \sum_j |u_j|^2$$

- **The function** $p \mapsto C(p)$ is a linear function!