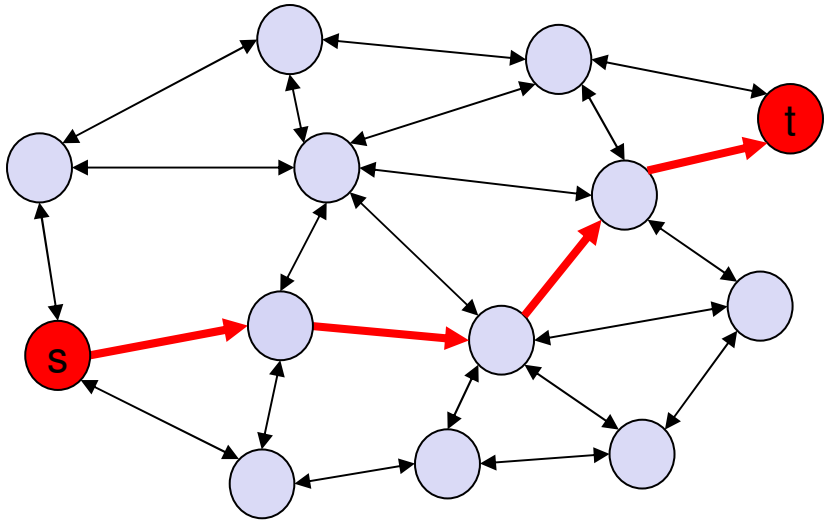
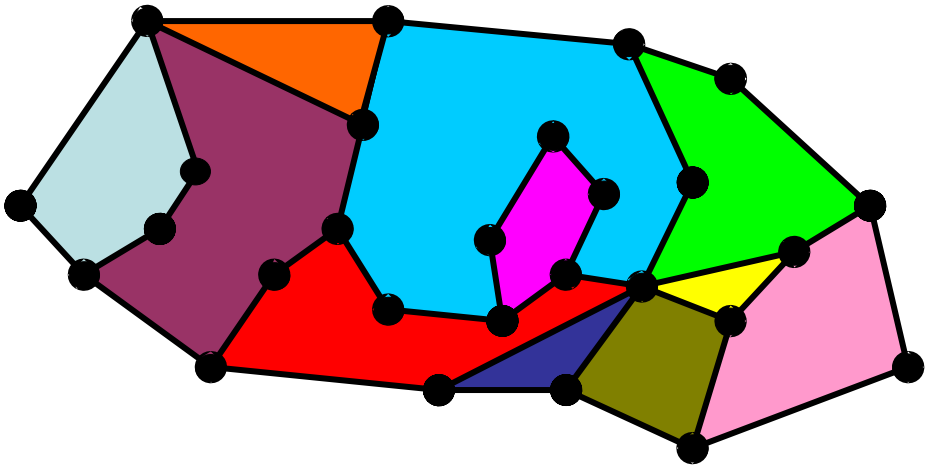
Routing with virtual coordinates

04/17/06

Overview of location-based routing

- Greedy routing + face routing on a planar subgraph.





Geographical forwarding

- How robust is geographical forwarding to location errors?
  - Accurate location information is hard to obtain.
- Face routing is problematic in practice.
  - Overload nodes on the boundaries of “holes”.
- Hope to establish “virtual coordinates” so that greedy routing does not get stuck.

Papers

Virtual coordinates:

- Ananth Rao, Christos Papadimitriou, Scott Shenker, and Ion Stoica, **Geographical routing without location information**, Proc. MobiCom'03, pages 96 - 108, 2003.
- James Newsome, Dawn Song, **GEM: Graph EMbedding for Routing and Data-Centric Storage in Sensor Networks Without Geographic Information**, Proc. Sensys'03.

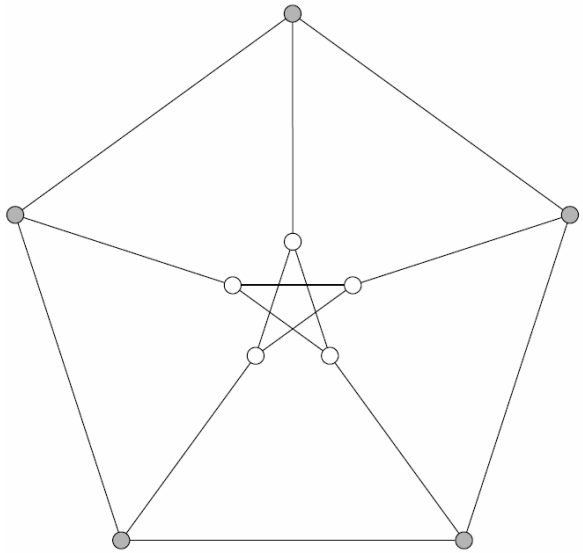
Approach I:
Rubber band representation

Goal: find a good layout of the network

- Let's just try to find a reasonable layout of the network.
- What is a good layout?
- Stretch the network out.
- Then try the geographical routing on this "virtual" coordinates.

Rubber band drawing of a graph

- All edges are rubber bands.
- Nail down some nodes S in the plane, let the graph go.
- Theorem: the algorithm converges to a unique state – rubber band representation extending S.
- Recall the mass-spring model in localization…



Peterson graph with one pentagon nailed down.

Rubber band drawing of a graph

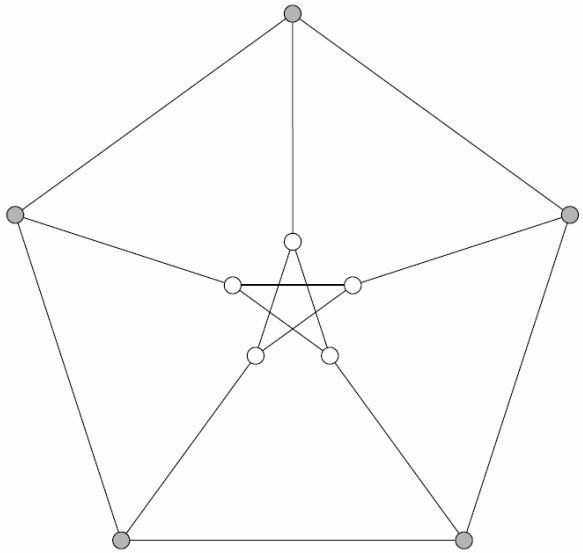
- The rubber band algorithm minimizes the total energy:

$$\mathcal{E}(x) = \sum_{ij \in E} |x_i - x_j|^2$$

- Claim: E(x) is convex.

$$\mathcal{E}(x) = \sum_{ij \in E} \sum_{k=1}^{d} (x_{ik} - x_{jk})^2.$$

- When any x_i goes to infinity, E(x) goes to infinity. So we have a unique global minimum.



Peterson graph with one pentagon nailed down.

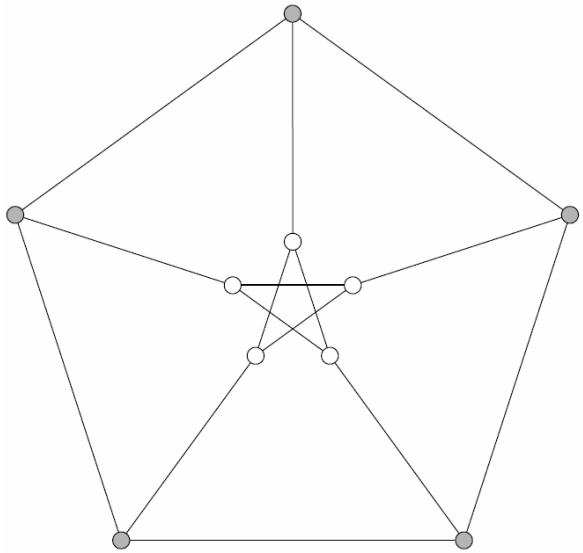
Rubber band drawing of a graph

- How does the rubber band representation look like?
- $\partial E(x)/\partial x_i = 0$.

$$\sum_{j \in N(i)} (x_i - x_j) = 0.$$

neighbors

- The rubber band connecting i and j pulls i with force $x_j - x_i$. The total force acting on x_i is 0.
- The graph is at equilibrium.



Peterson graph with one pentagon nailed down.

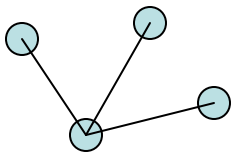
Rubber band drawing of a graph

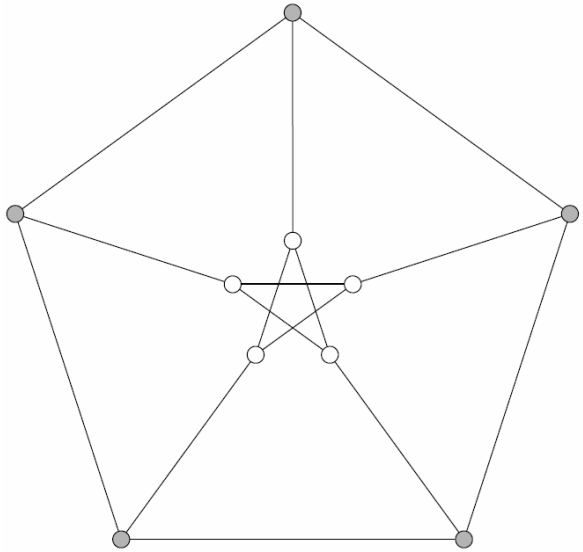
$$\sum_{j \in N(i)} (x_i - x_j) = 0.$$

1. Every free node is at the center of gravity of its neighbors.

$$x_i = \frac{1}{d_i} \sum_{j \in N(i)} x_j.$$

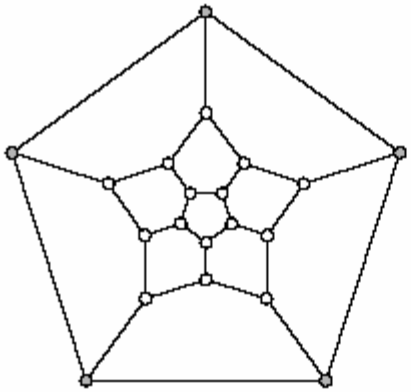
2. no reflex vertices.

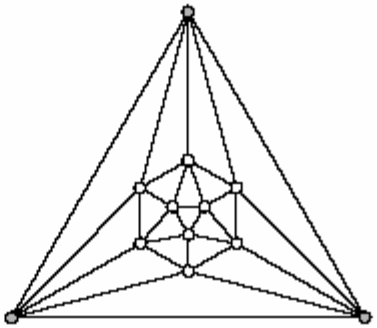


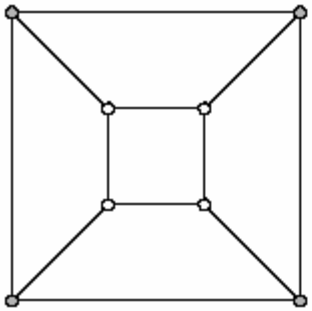


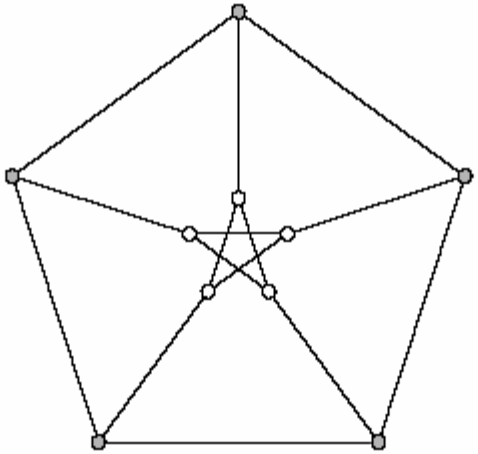
Peterson graph with one pentagon nailed down.

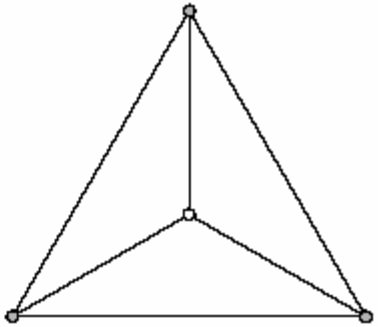
More examples











Rubber band algorithm

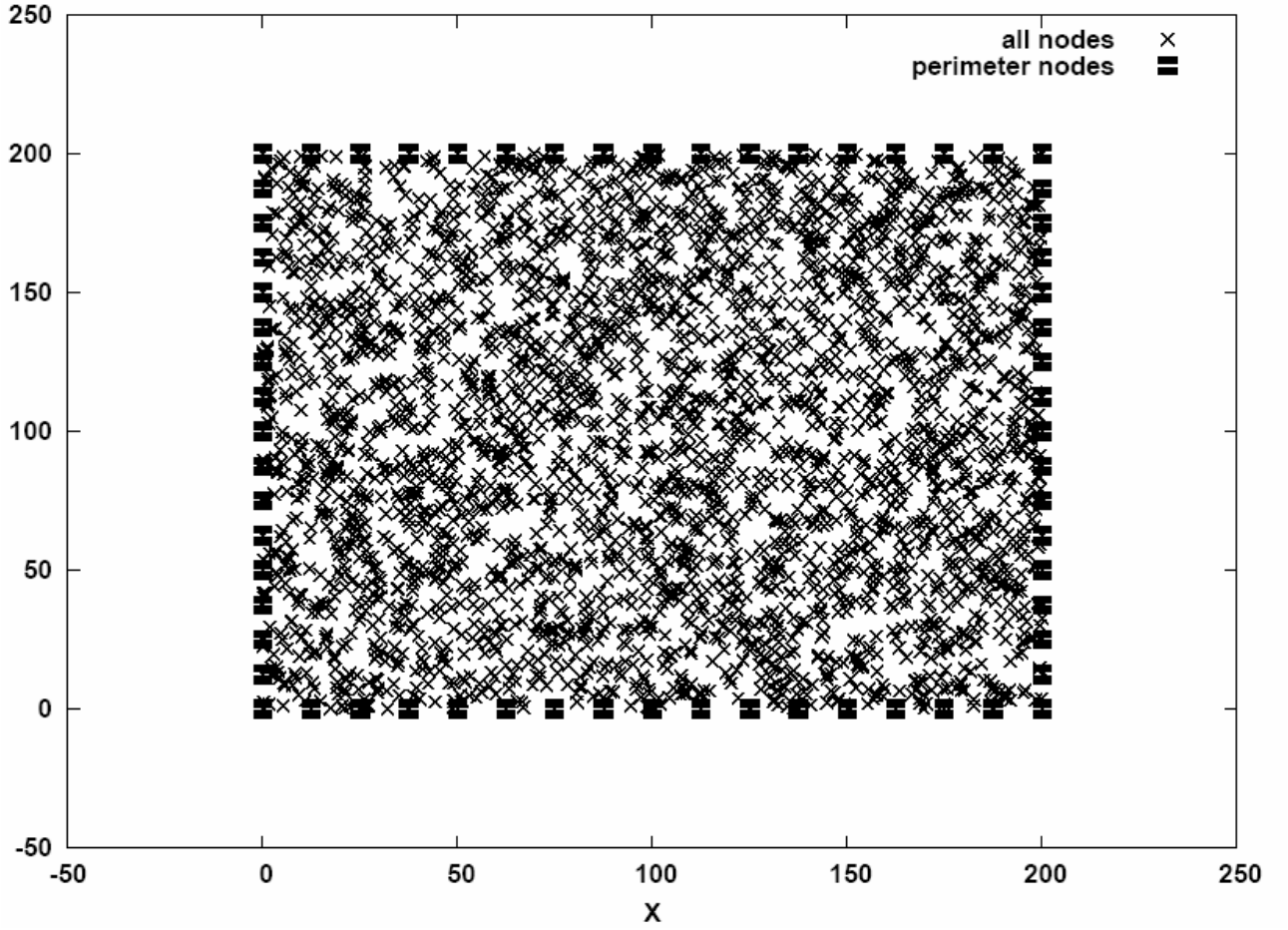
- Recall the mass-spring model.
- First we assume nodes on the boundary know their location.
- Fix the nodes on the outer boundary.
- Iterative algorithm:
  - Every node moves to the center of gravity of its neighbors.

$$x_i \Leftarrow \frac{1}{d_i} \sum_{j \in N(i)} x_j.$$

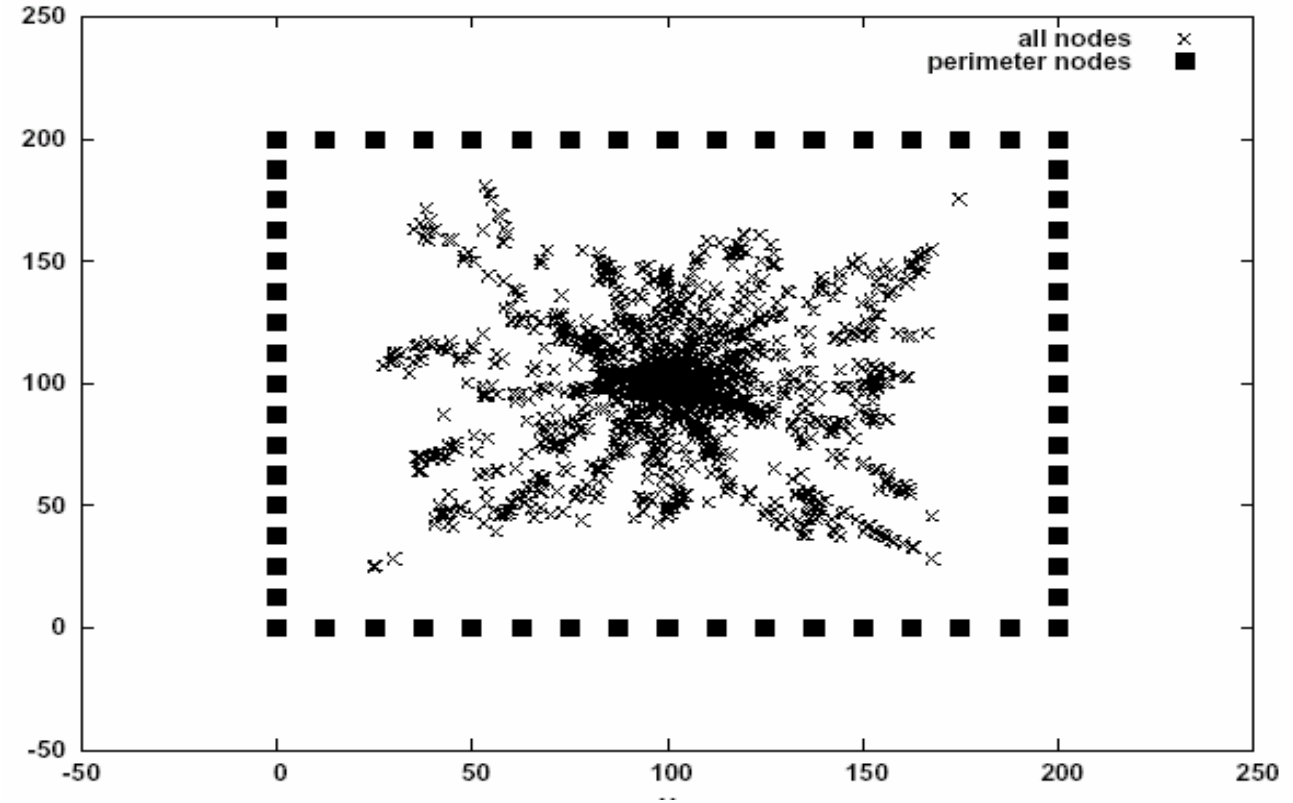
- Until no node moves more than distance δ.

A network with 3200 nodes

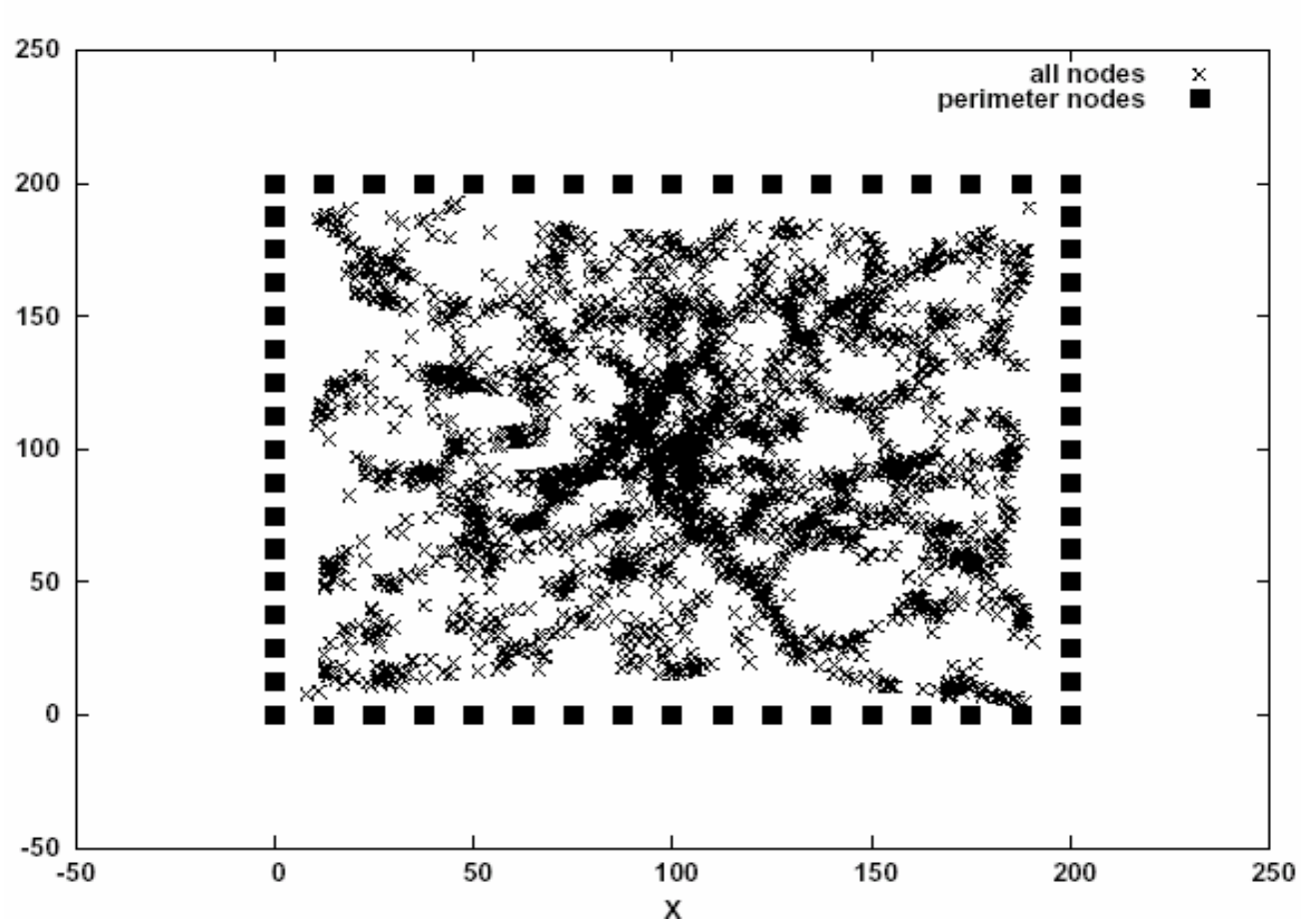
- Greedy routing success rate: 0.989, avg path length 16.8



Perimeter nodes are known (10 iterations)

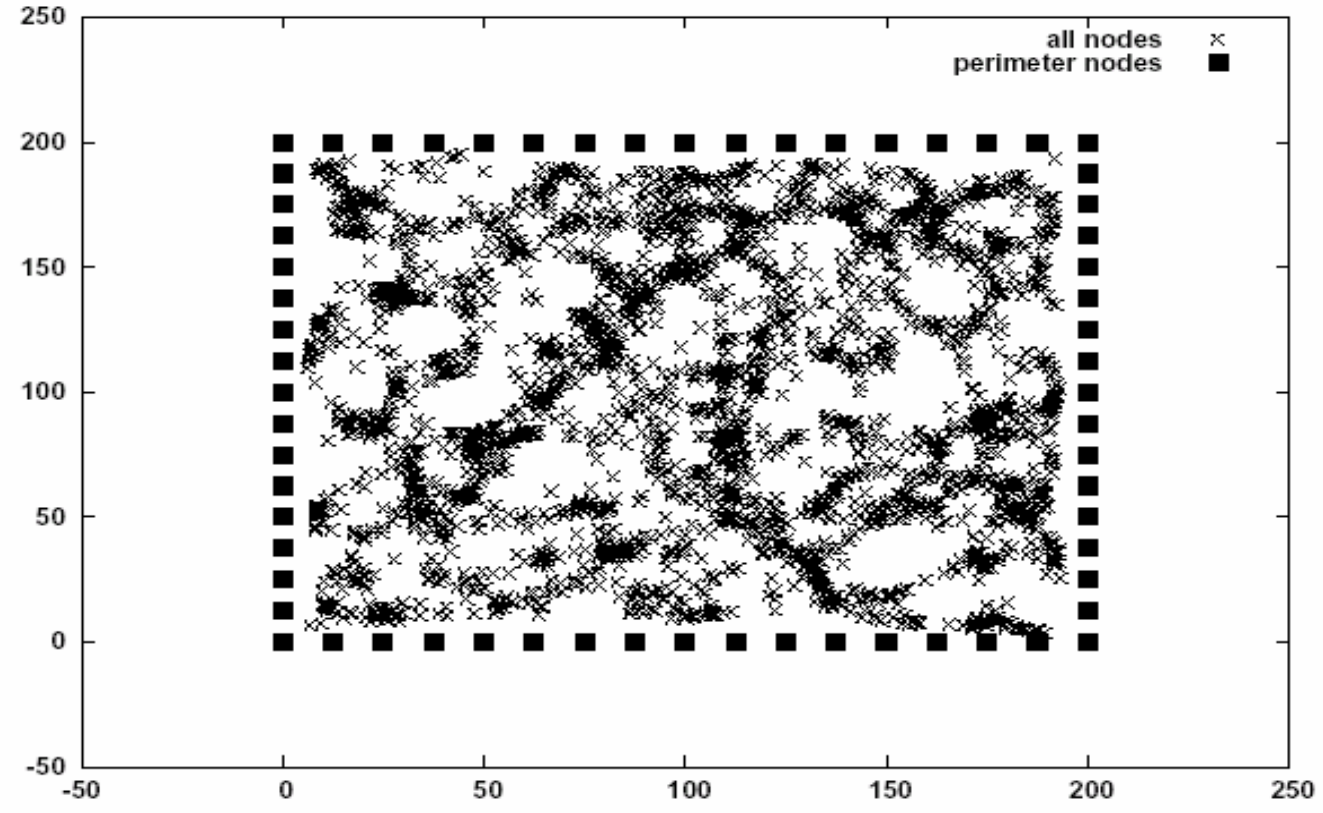


Perimeter nodes are known (100 iterations)



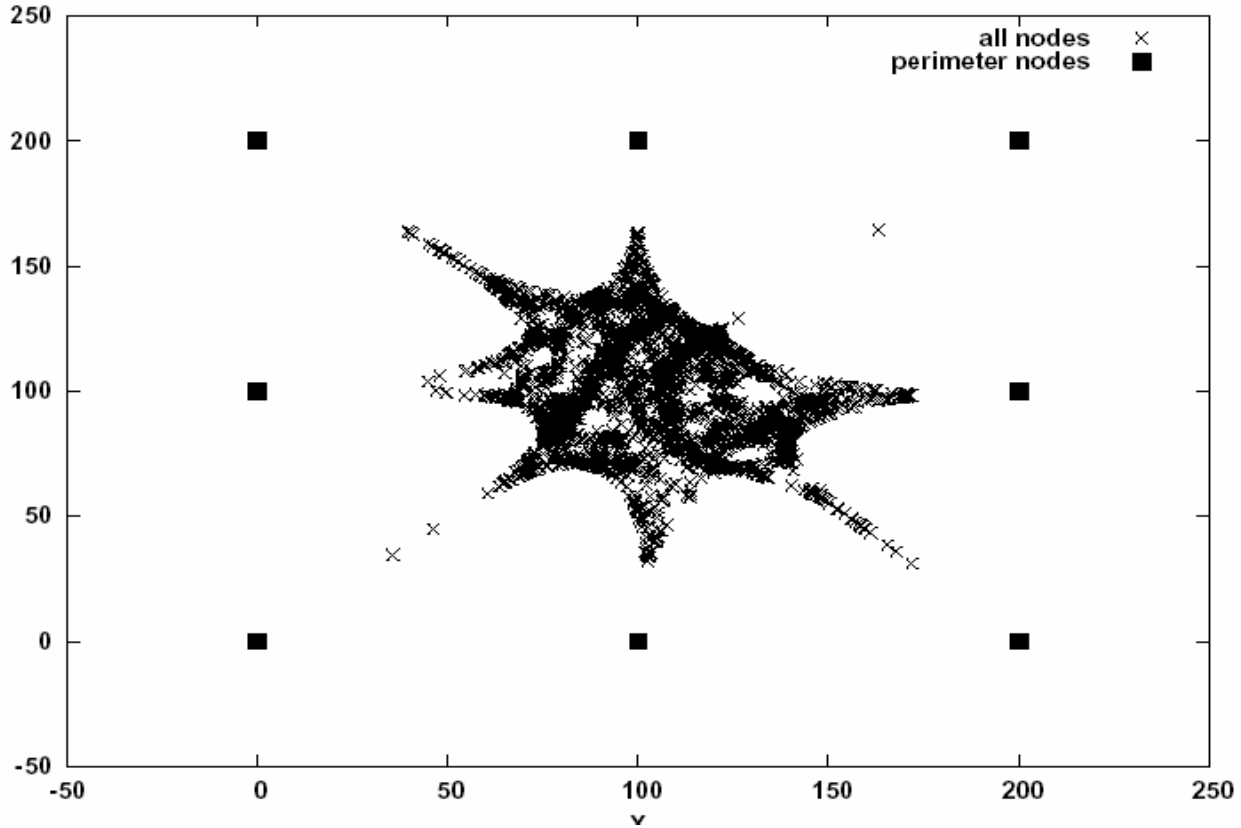
Perimeter nodes are known (1000 iterations)

- Greedy routing success rate: 0.993, avg path length 17.1



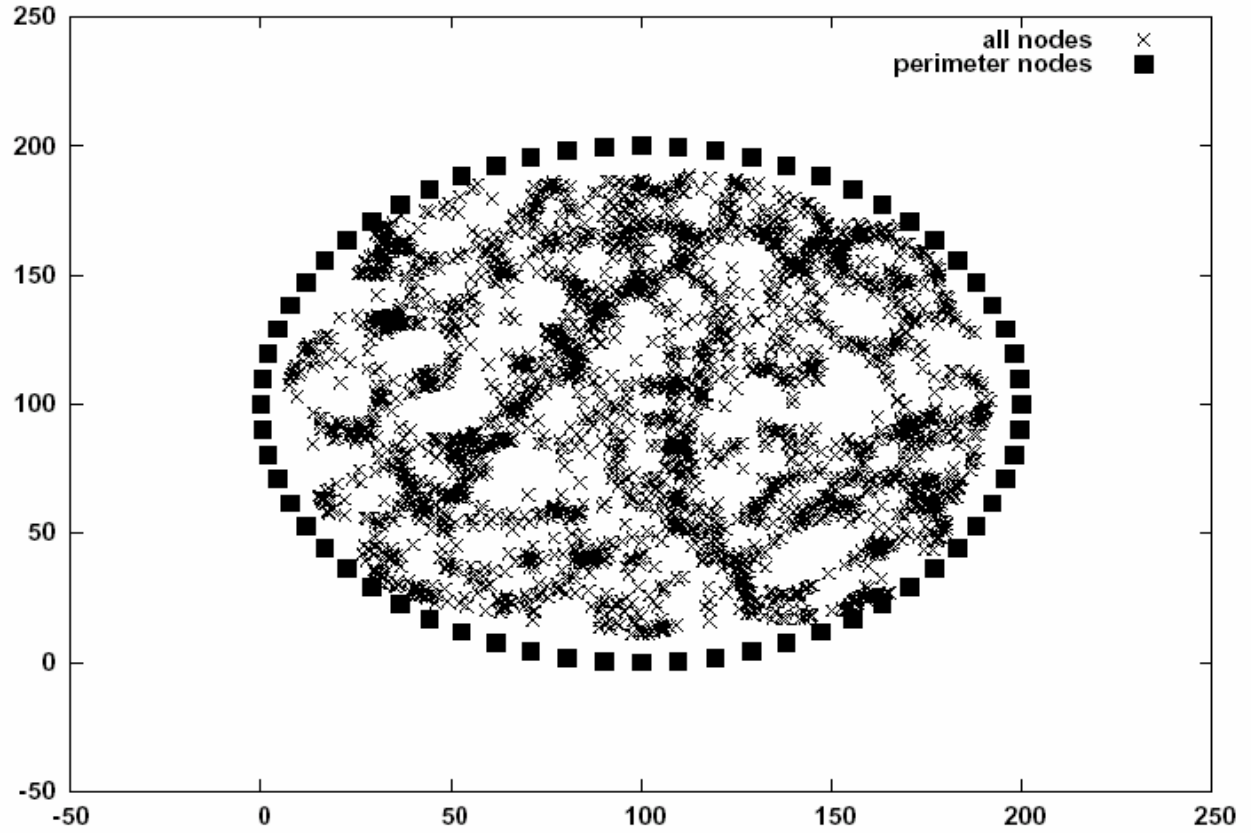
Resiliency of the rubber band approach

- Greedy routing success rate: 0.981, avg path length 17.3



Resiliency of the rubber band approach

- Greedy routing success rate: 0.99, avg path length 17.1



Perimeter nodes

- Need nodes on the perimeter to “stretch” out the net.

- First assume we know nodes on the perimeter, but not the locations.

1. Each perimeter sends hello messages.
2. All the nodes record hop counts to each perimeter node.
3. The hop count between every pair of perimeter node is broadcast to all perimeter nodes.
4. Embed perimeter nodes in the plane.

$$\sum_{i,j \in perimeter\_set} (measured\_dist(i,j) - dist(i,j))^2$$

Perimeter nodes

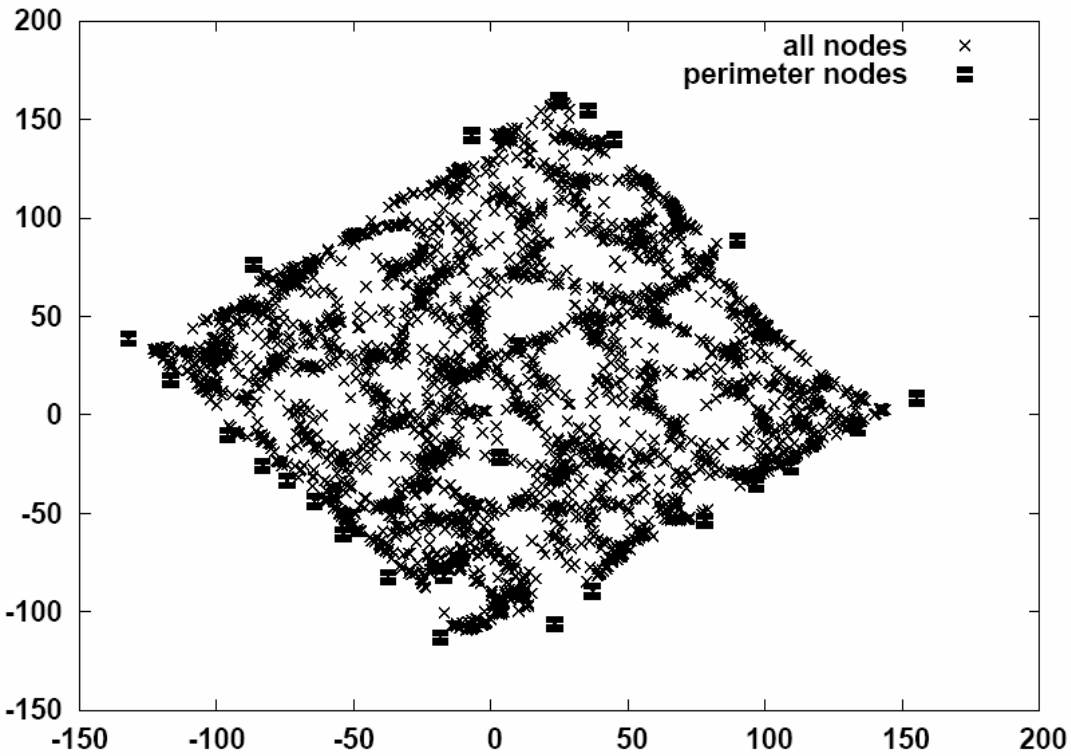
1. The embedding only gives relative positions: include 2 bootstrapping beacons in the embedding of perimeters.
   - Use the center of gravity as origin.
   - 1st bootstrap node defines the positive x-axis.
   - 2nd bootstrap node defines the positive y-axis.

2. Non-perimeter nodes actually have the distances to all perimeter nodes. So they can also embed themselves.
   - Gives good initial positions for the rubber band algorithm.

3. If a perimeter node doesn't receive the pairwise hop counts among all perimeter nodes, then it is not counted in the calculation.

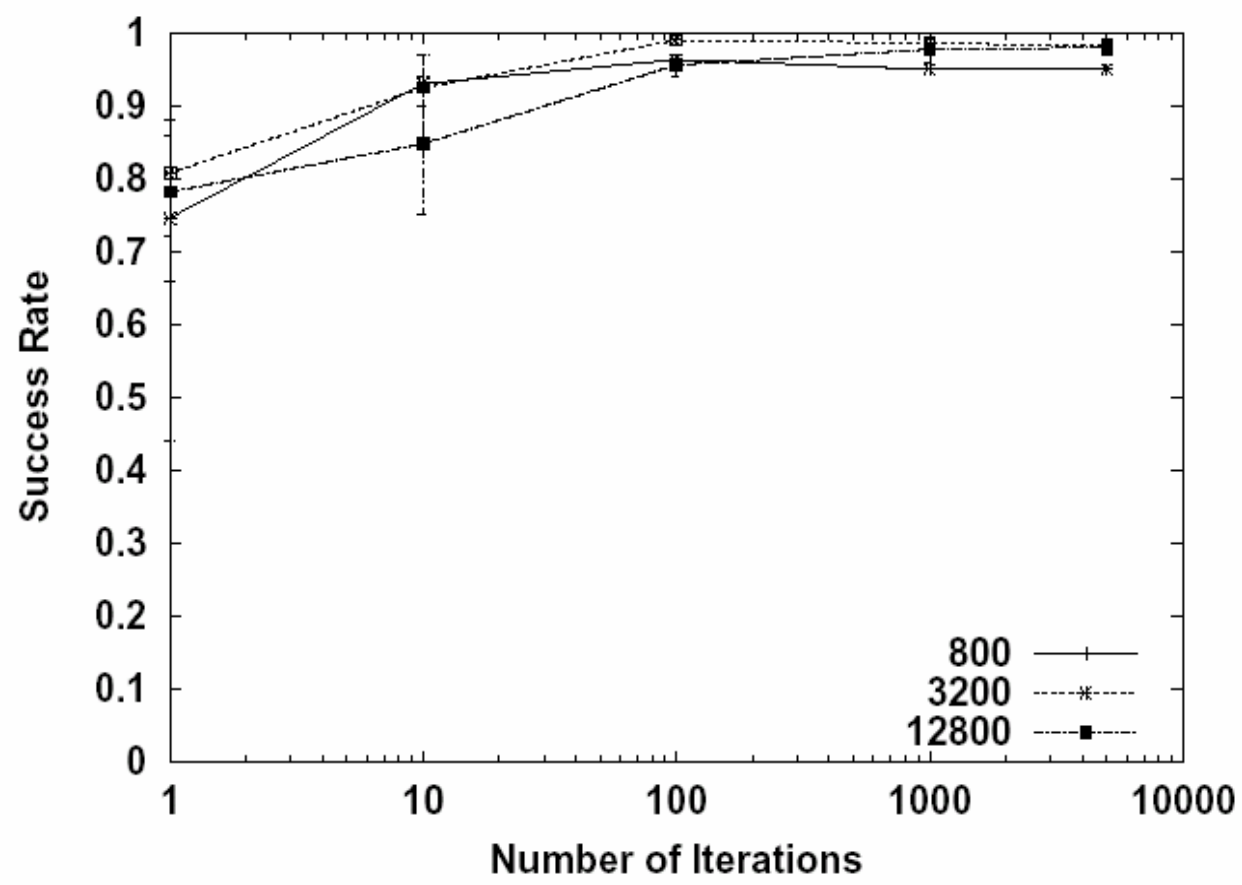
How to find perimeter nodes?

- The bootstrapping nodes send hello messages to everyone.
- The node which is the farthest among all its 2-hop neighbors will identify itself as a perimeter node.

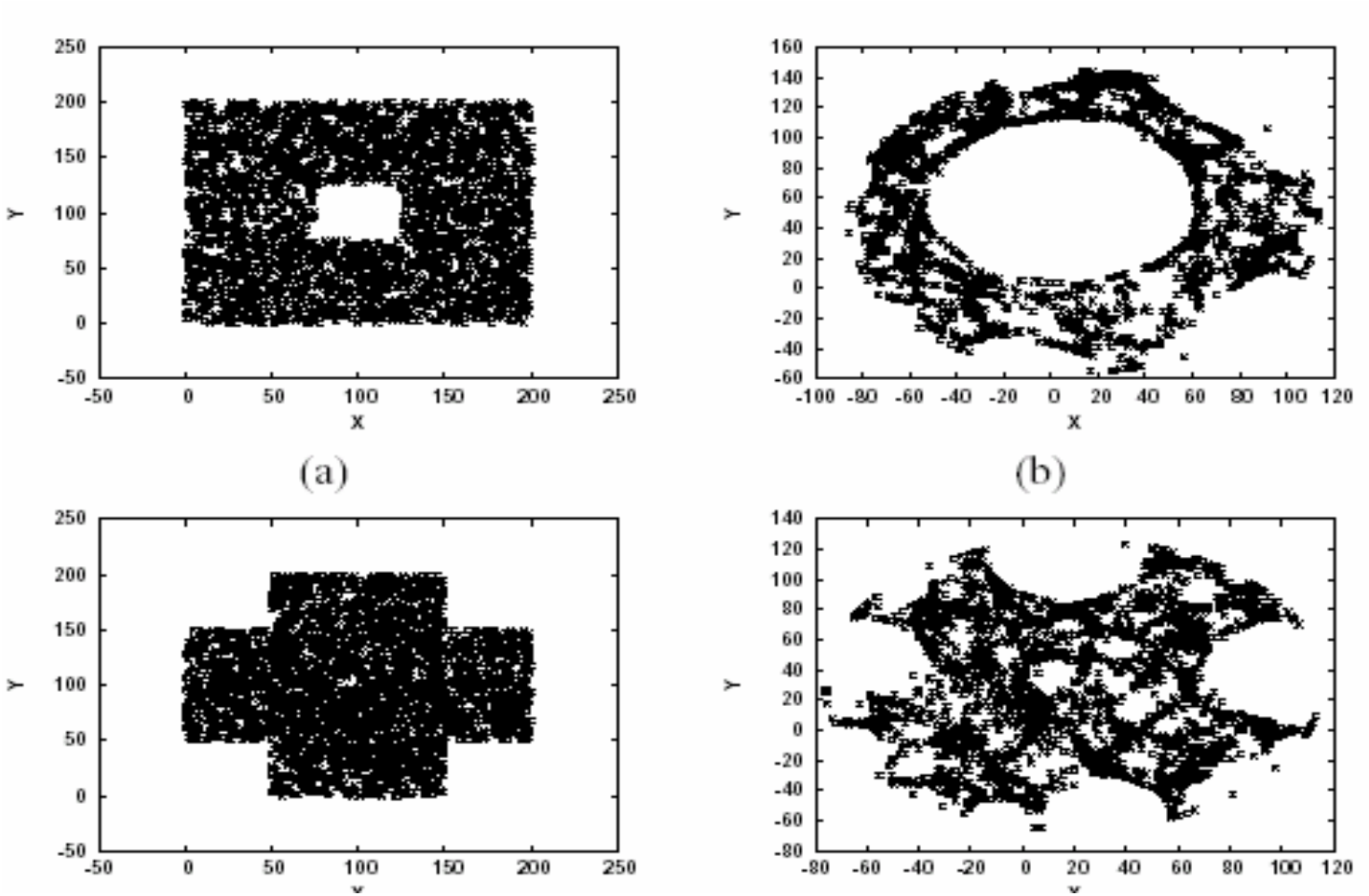


Success rate of greedy routing

- Success rate on virtual coordinates is comparable with true coordinates, when the sensors are dense and uniform.

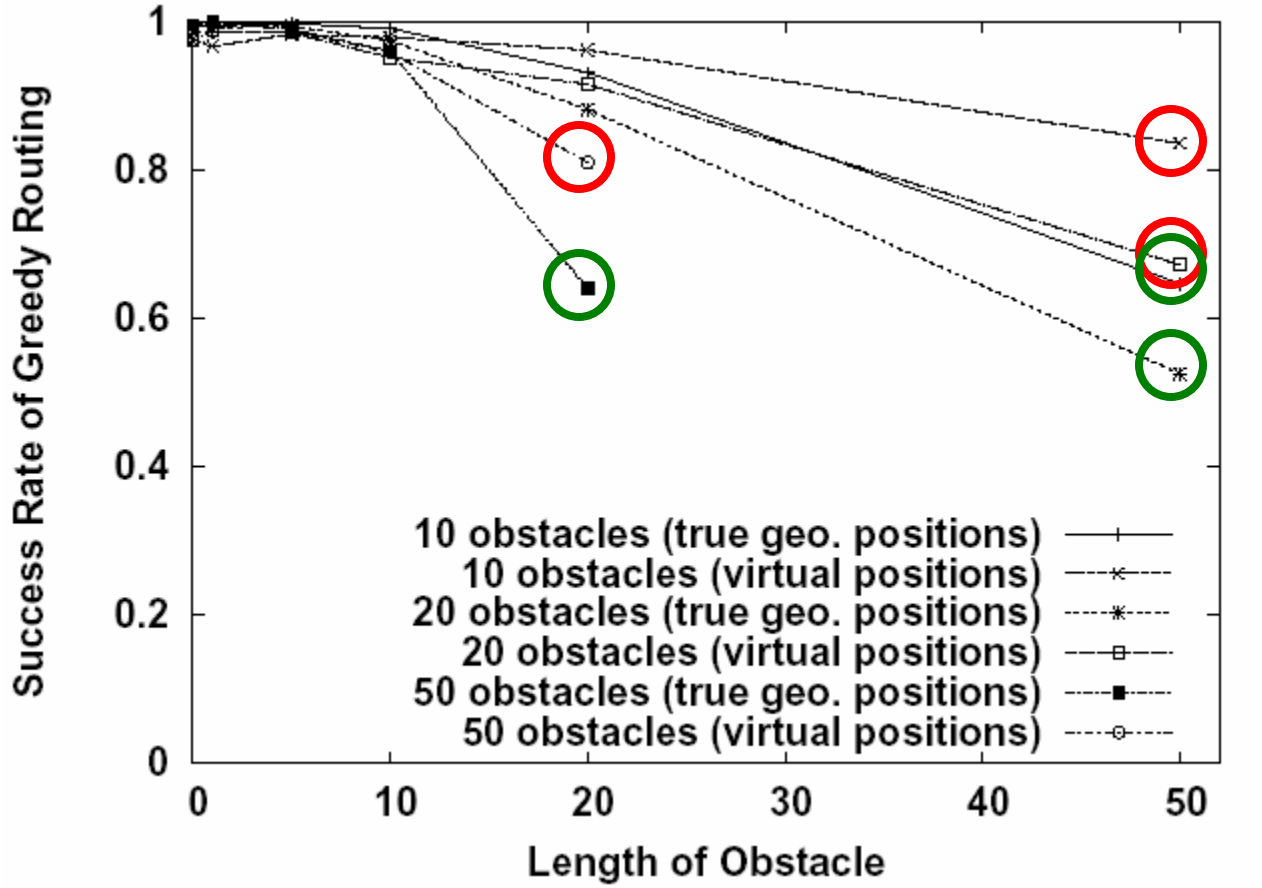


Weird Shapes



Obstacles

- Success rate on virtual coordinates degrades when there are a lot of obstacles, but better than true coordinates.



Conclusions

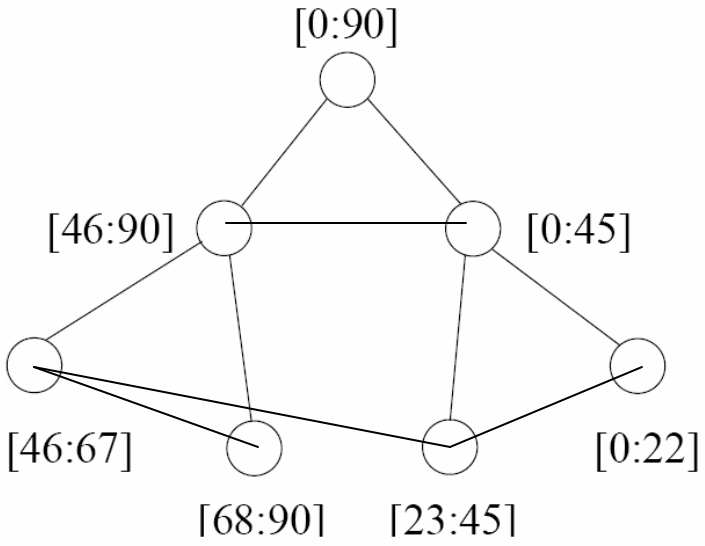
- Geographical forwarding is quite robust to localization errors, or reasonable virtual coordinates.
- Geographical forwarding can easily scale to tens of thousands of nodes with acceptable overhead.
- For dense uniform sensor layout, we can eliminate the need for face routing altogether.

- Virtual coordinates respect the connectivity better than the true coordinates.

Approach II:
Embed a spanning tree in polar coordinate system

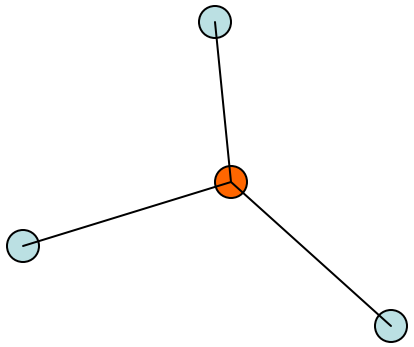
Embed a tree in polar coordinate system

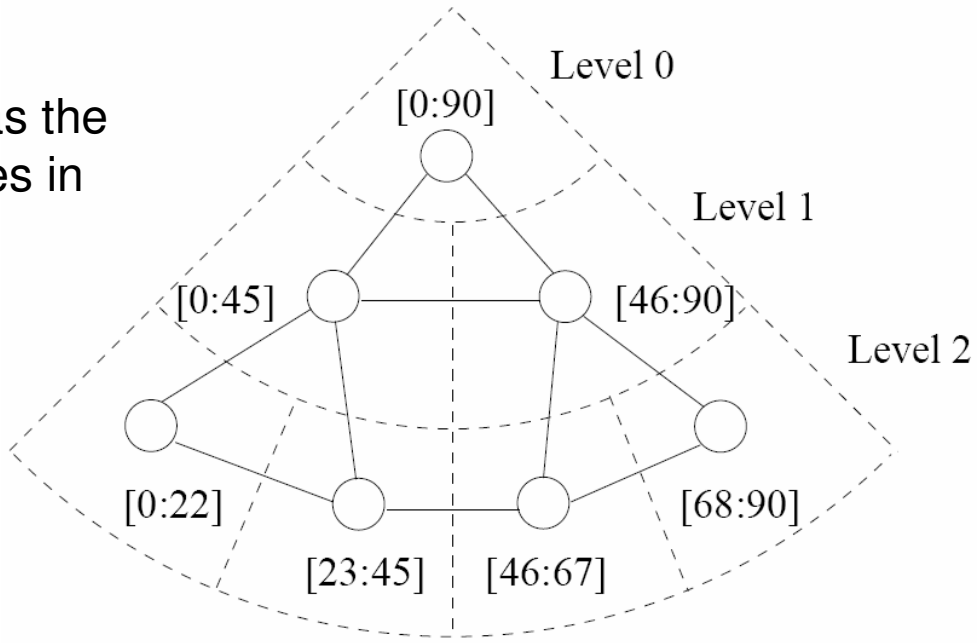
- Start from any node as root, flood to find the shortest path tree.
- Assign polar ranges to each node in the tree.
  - The range of a node is divided among its children.
  - The size of the range is proportional to the size of its subtree.
- Order the subtrees that align with the sensor connectivity.



Embed a tree in polar coordinate system

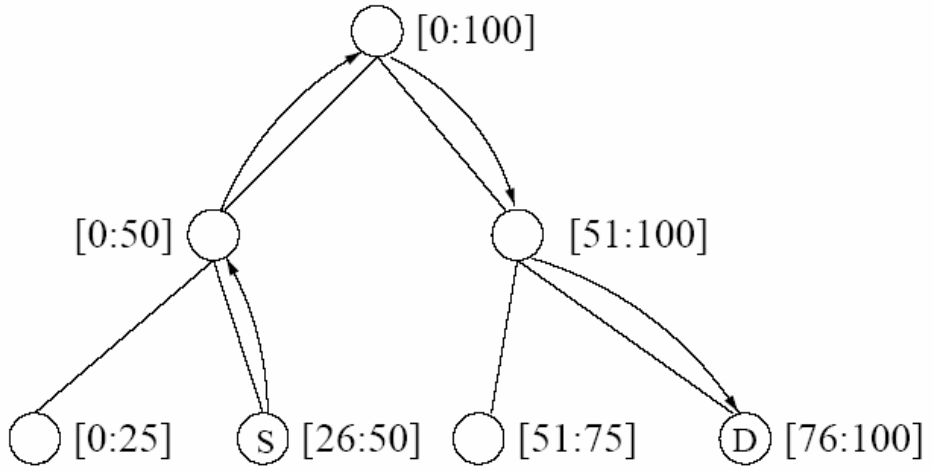
- Order the subtrees that align with the sensor connectivity.
  - Three reference nodes flood the network. Each node knows the hop count to each reference.
  - Each node embed itself with respect to the references.
  - A node's position is defined as the center of mass of all the nodes in its subtree.
  - This will provide an angular ordering of all the children.





Routing on a tree

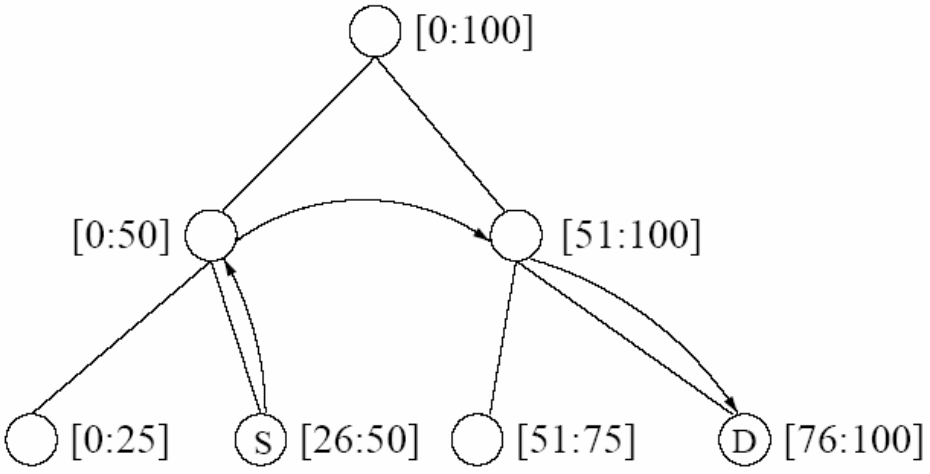
- Route to the common ancestor of the source and destination.
  - Check whether the destination range is included in the range of the current node.
  - If not, go to the parent.
  - Otherwise go to the corresponding child.
- Root is the bottleneck
- Path may be long.

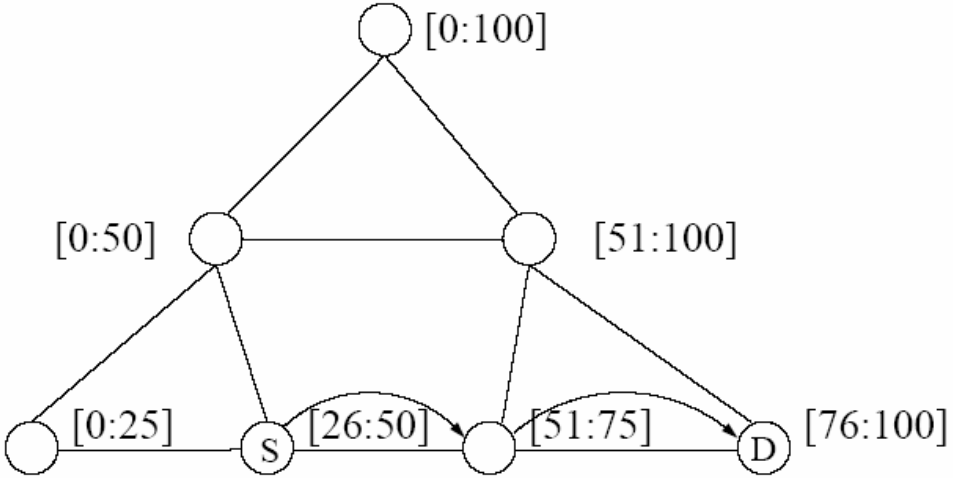


Routing on a tree

- Be a little smarter: store a local routing table that keeps the ranges of up to k-hop neighbors. ➔ find shortcuts.
- Virtual Polar Coordinate Routing: check the neighborhood, find the node that is closer to the destination.

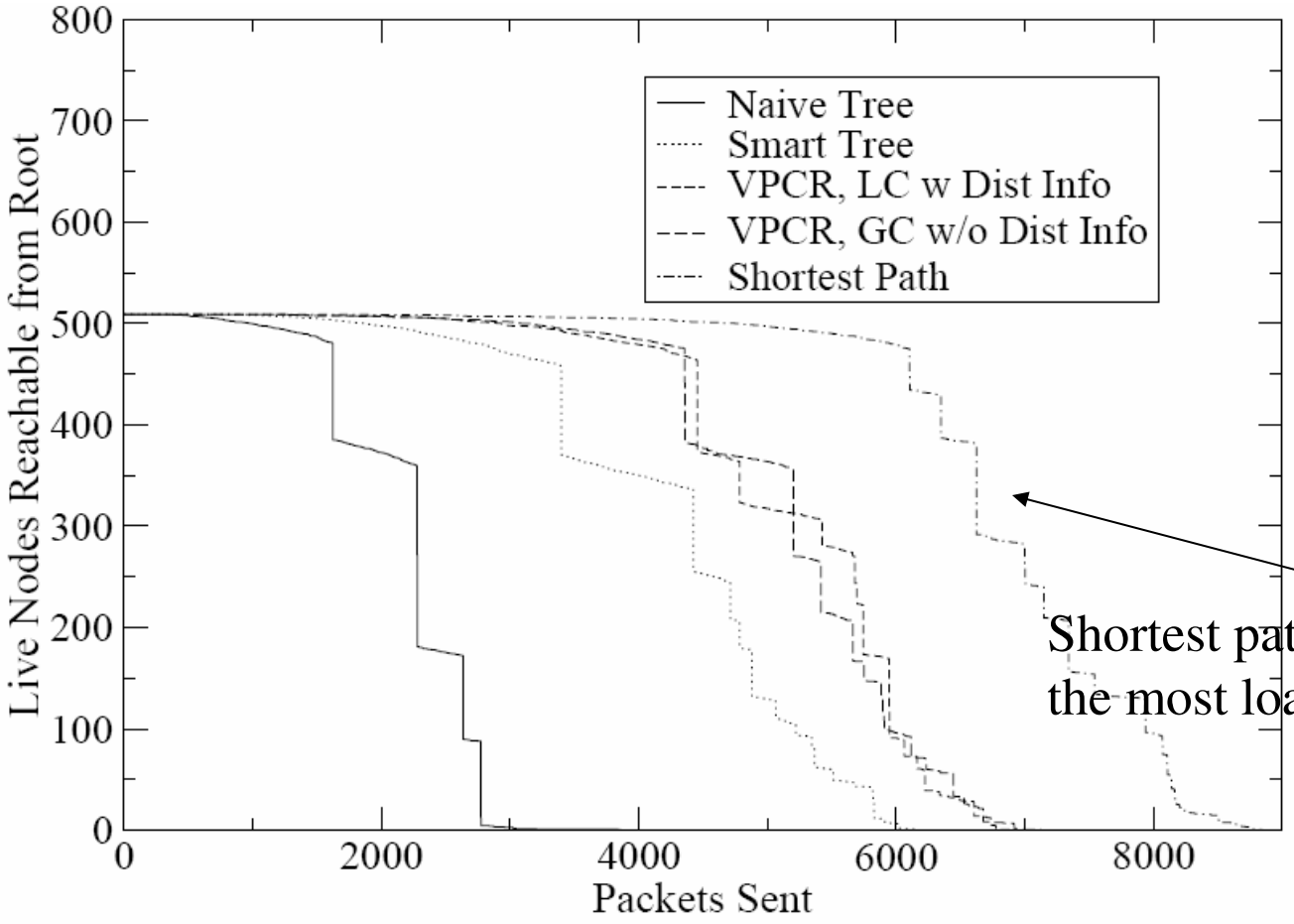
If the upper/lower bound is closer to the destination.





Load balancing

- Root is still the bottleneck even for smart routing.



Shortest path routing, still not the most load balanced routing

Summary

- Replace the “real” coordinates by “virtual” coordinates for routing.
- Goal: no need for accurate location information.