Chapter 6
GEOMETRIC ROUTING

Distributed
Computing
Group

Mobile Computing
Winter 2005 / 2006

Overview – Geometric Routing

- Geometric routing
- Greedy geometric routing

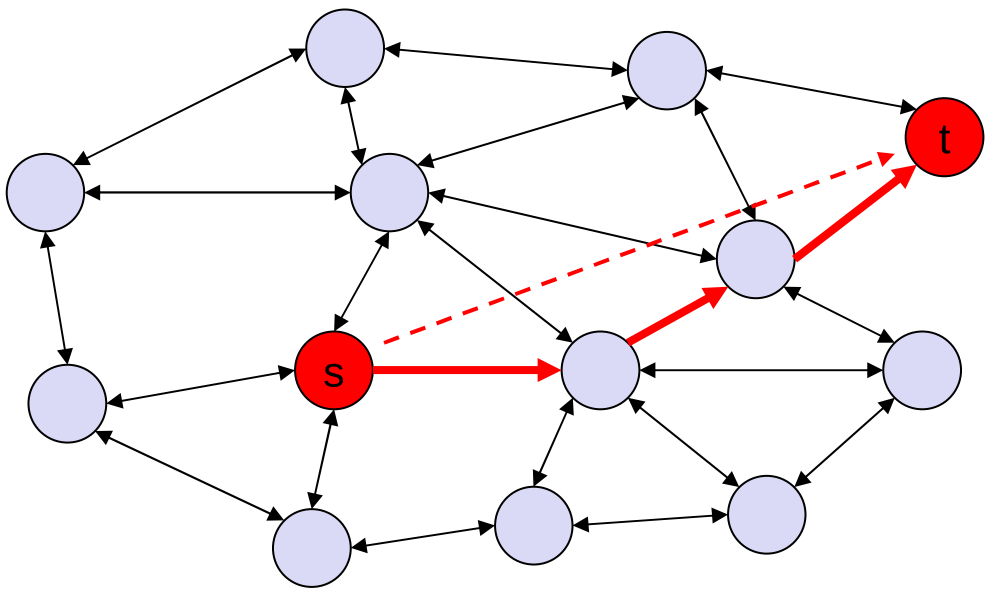
- Euclidean and planar graphs
- Unit disk graph
- Gabriel graph and other planar graphs

- Face Routing
- Greedy and Face Routing

- Geometric Routing without Geometry

Geometric (geographic, directional, position-based) routing

- …even with all the tricks there will be flooding every now and then.
- In this chapter we will assume that the nodes are location aware (they have GPS, Galileo, or an ad-hoc way to figure out their coordinates), and that we know where the destination is.
- Then we simply route towards the destination

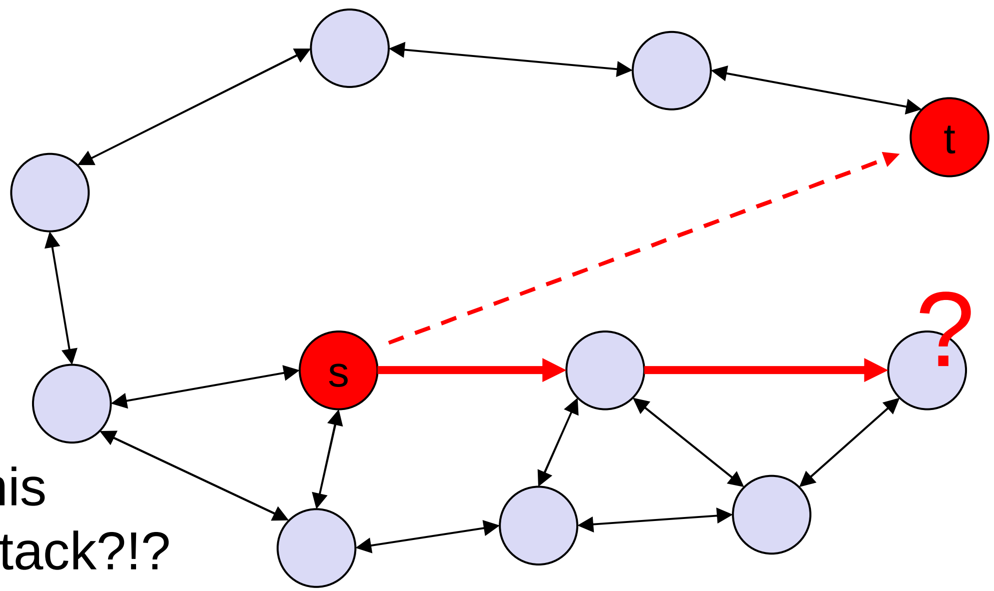


Geometric routing

- Problem: What if there is no path in the right direction?

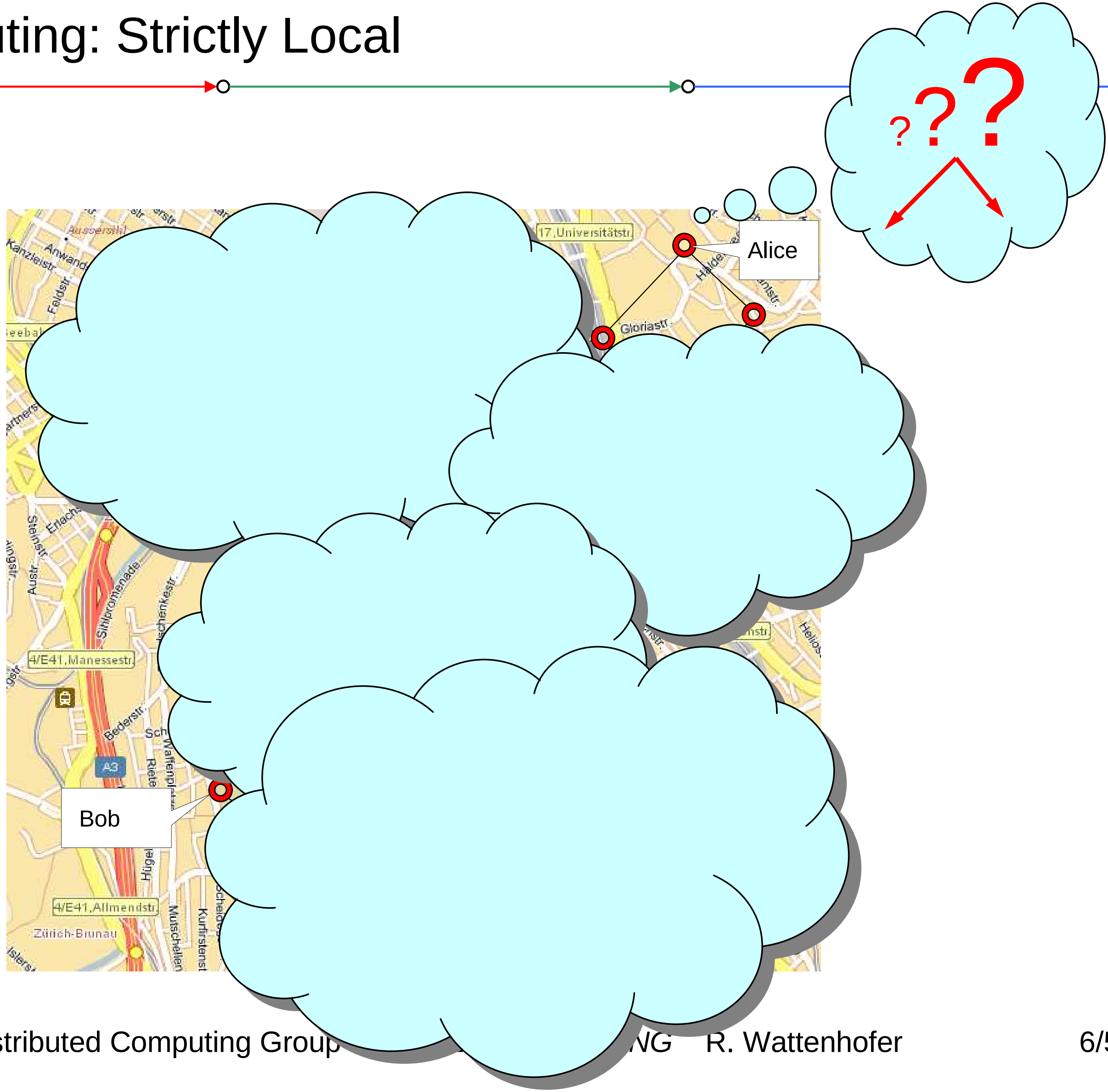
- We need a guaranteed way to reach a destination even in the case when there is no directional path…

- Hack: as in flooding nodes keep track of the messages they have already seen, and then they backtrack* from there



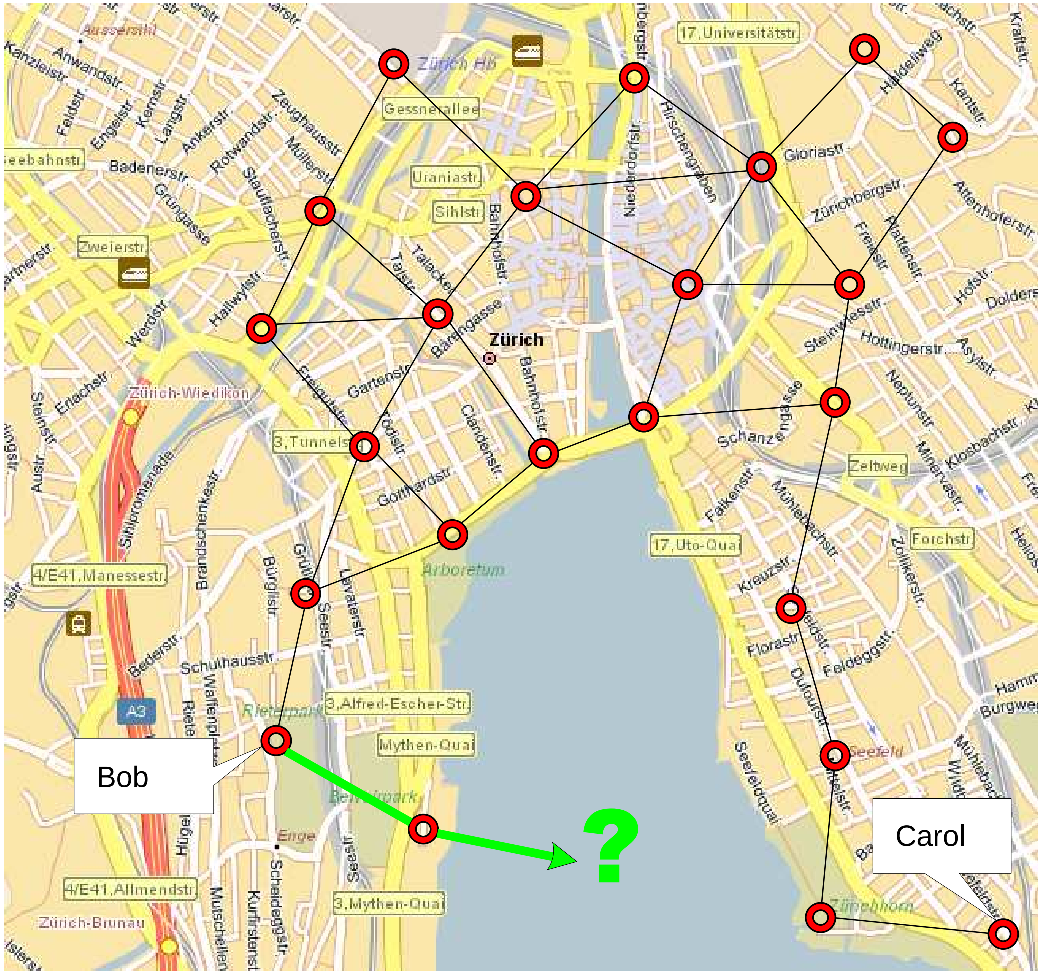
*backtracking? Does this mean that we need a stack?!?

Geo-Routing: Strictly Local



Greedy Geo-Routing?

Greedy Geo-Routing?



What is Geographic Routing?

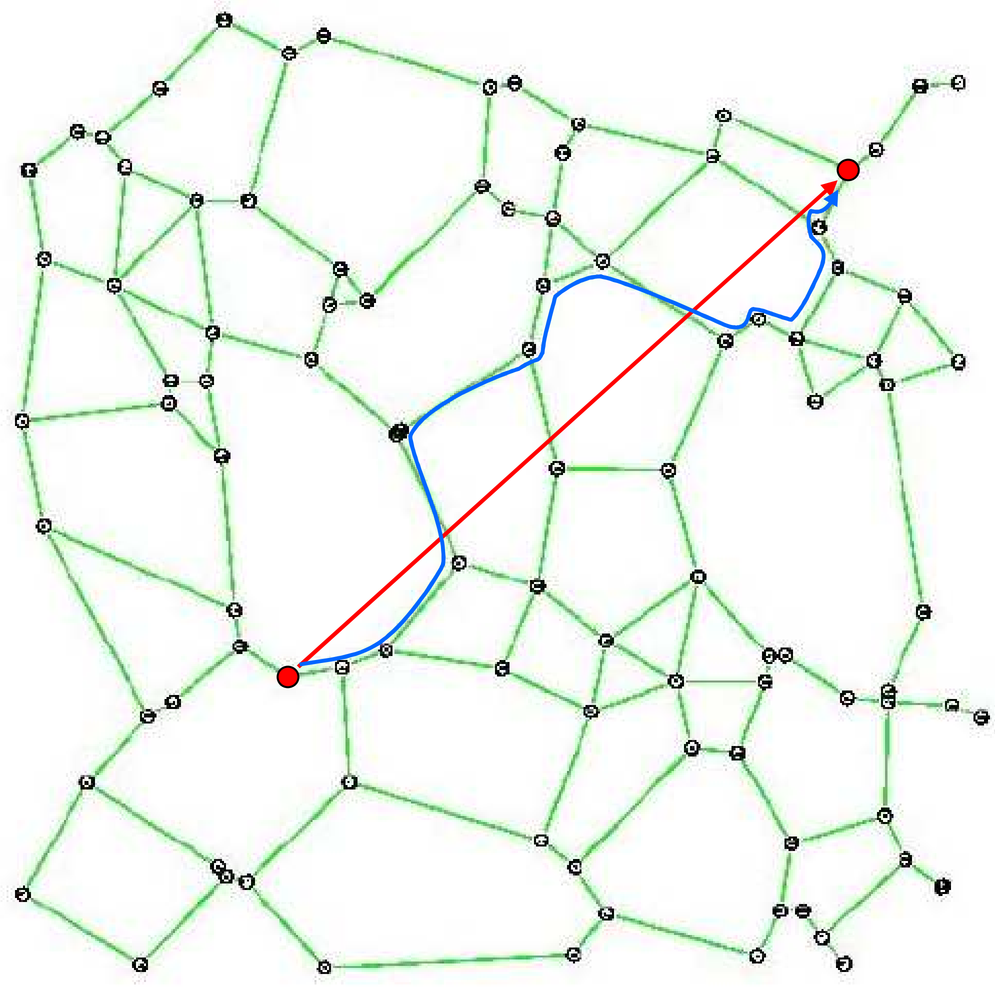
- A.k.a. geometric, location-based, position-based, etc.

- Each node knows its own position and position of neighbors
- Source knows the position of the destination
- No routing tables stored in nodes!

- Geographic routing makes sense
  - Own position: GPS/Galileo, local positioning algorithms
  - Destination: Geocasting, location services, source routing++
  - Learn about ad-hoc routing in general

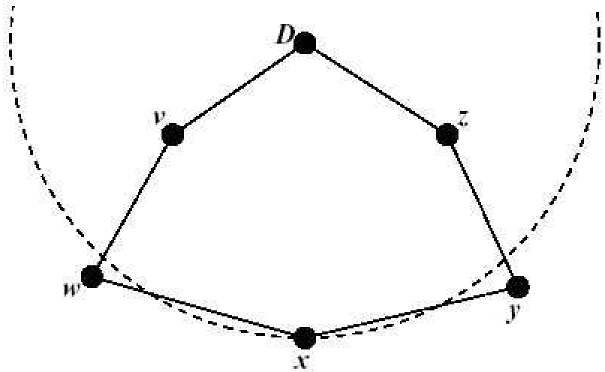
Greedy routing

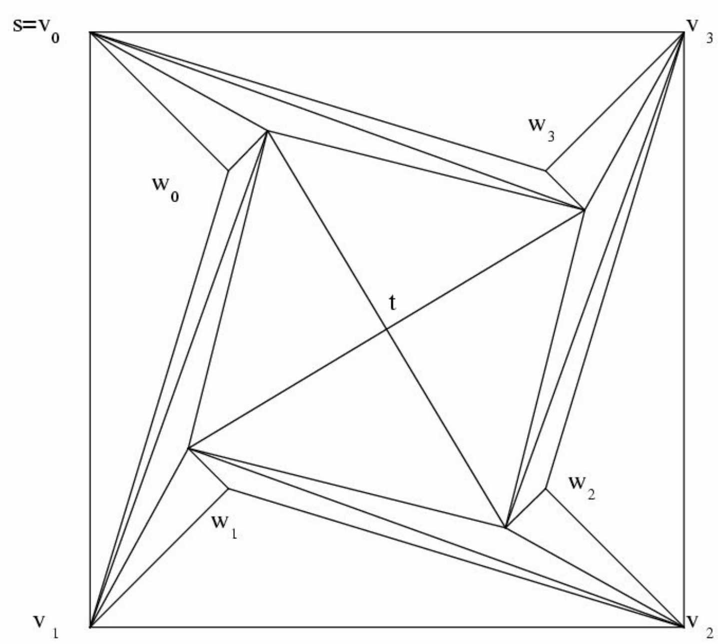
- Greedy routing looks promising.
- Maybe there is a way to choose the next neighbor and a particular graph where we always reach the destination?



Examples why greedy algorithms fail

- We greedily route to the neighbor which is closest to the destination: But both neighbors of x are not closer to destination D

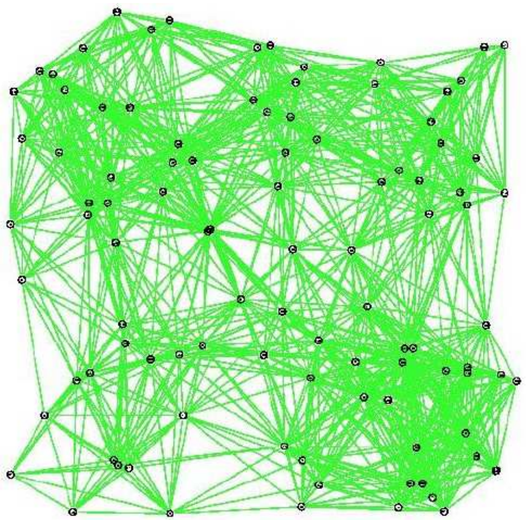


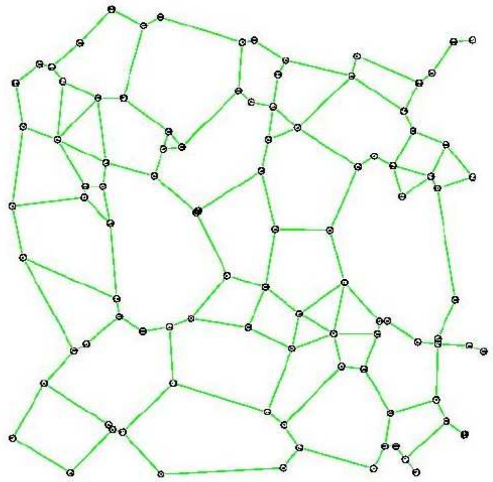


- Also the best angle approach might fail, even in a triangulation: if, in the example on the right, you always follow the edge with the narrowest angle to destination t, you will forward on a loop $v_0, w_0, v_1, w_1, \ldots, v_3, w_3, v_0, \ldots$

Euclidean and Planar Graphs

- Euclidean: Points in the plane, with coordinates
- Planar: can be drawn without "edge crossings" in a plane



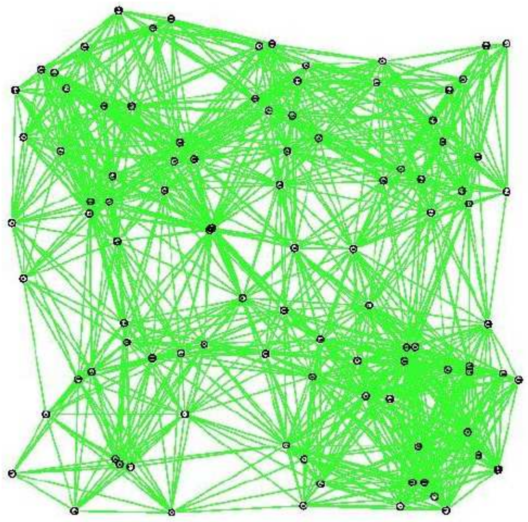


- Euclidean planar graphs (planar embeddings) simplify geometric routing.

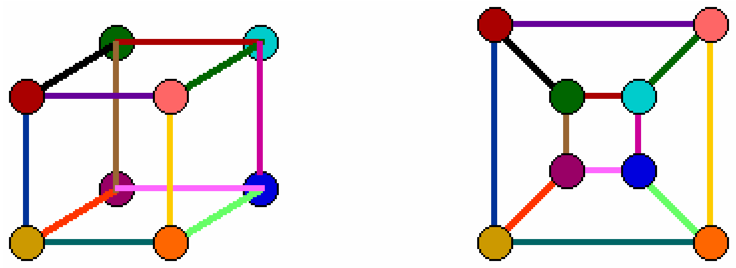
Unit disk graph

- We are given a set *V* of nodes in the plane (points with coordinates).
- The unit disk graph *UDG*(*V*) is defined as an undirected graph (with *E* being a set of undirected edges). There is an edge between two nodes *u*,*v* iff the Euclidean distance between *u* and *v* is at most 1.
- Think of the unit distance as the maximum transmission range.

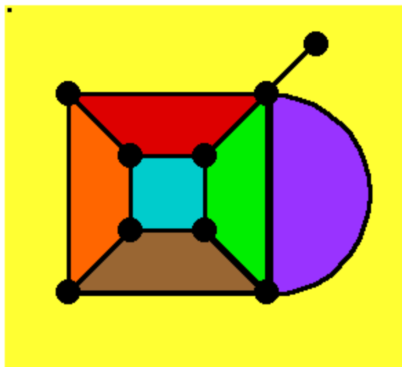
- We assume that the unit disk graph *UDG* is connected (that is, there is a path between each pair of nodes)
- The unit disk graph has many edges.
- Can we drop some edges in the *UDG* to reduced complexity and interference?



Planar graphs

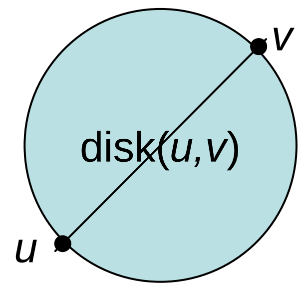


- Definition: A planar graph is a graph that can be drawn in the plane such that its edges only intersect at their common end-vertices.
- Kuratowski's Theorem: A graph is planar iff it contains no subgraph that is edge contractible to K_5 or $K_{3,3}$.
- Euler's Polyhedron Formula: A connected planar graph with *n* nodes, *m* edges, and *f* faces has $n - m + f = 2$.
- Right: Example with 9 vertices,14 edges, and 7 faces (the yellow "outside" face is called the infinite face)
- Theorem: A simple planar graph with n nodes has at most 3n–6 edges, for $n \geq 3$.

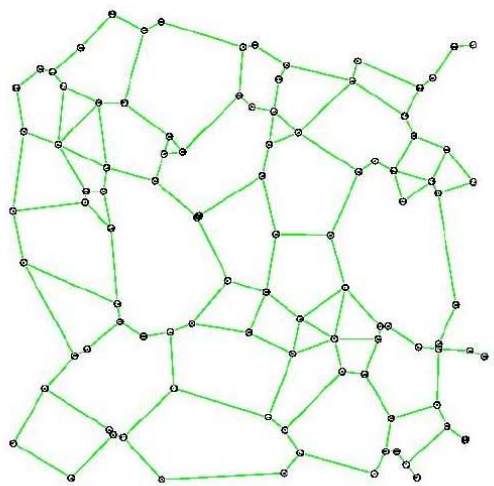


Gabriel Graph

- Let disk(*u*,*v*) be a disk with diameter (*u*,*v*) that is determined by the two points *u*,*v*.
- The Gabriel Graph GG(*V*) is defined as an undirected graph (with *E* being a set of undirected edges). There is an edge between two nodes *u*,*v* iff the disk(*u*,*v*) including boundary contains no other points.

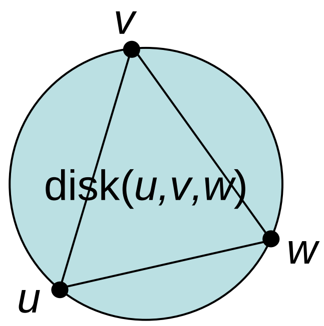


- As we will see the Gabriel Graph has interesting properties.

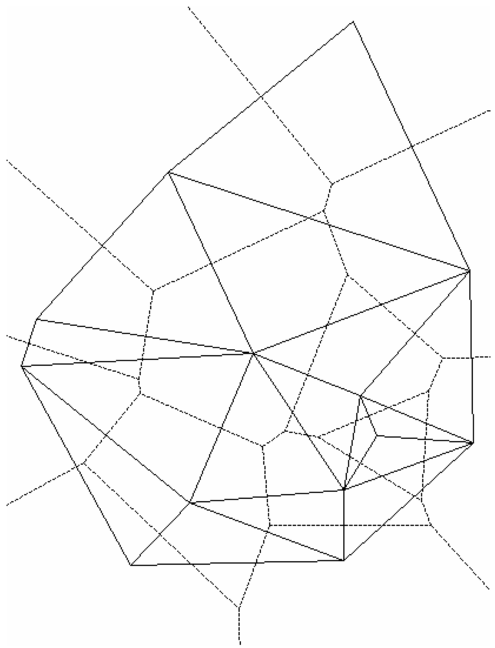


Delaunay Triangulation

- Let disk(*u,v,w*) be a disk defined by the three points *u*,*v,w*.
- The Delaunay Triangulation (Graph) DT(*V*) is defined as an undirected graph (with *E* being a set of undirected edges). There is a triangle of edges between three nodes *u*,*v,w* iff the disk(*u,v,w*) contains no other points.

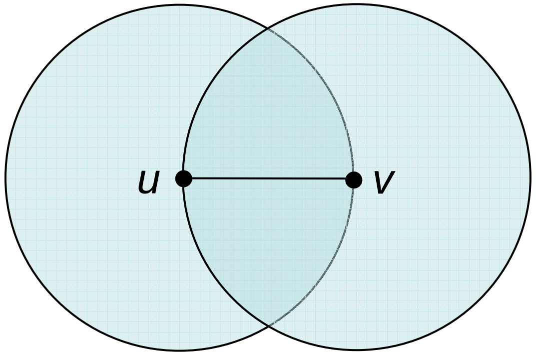


- The Delaunay Triangulation is the dual of the Voronoi diagram, and widely used in various CS areas; the DT is planar; the distance of a path (s,…,t) on the DT is within a constant factor of the s-t distance.

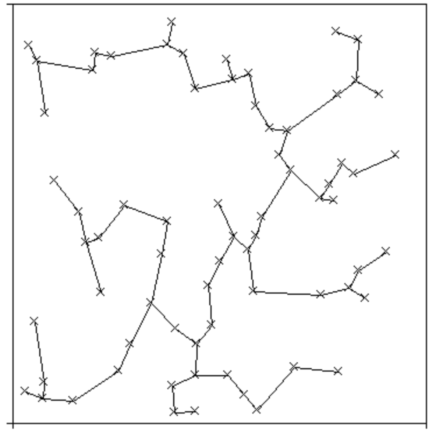


Other planar graphs

- Relative Neighborhood Graph RNG(V)
- An edge e = (u,v) is in the RNG(V) iff there is no node w with (u,w) < (u,v) and (v,w) < (u,v).



- Minimum Spanning Tree MST(V)
- A subset of *E* of *G* of minimum weight which forms a tree on *V*.



Properties of planar graphs

- Theorem 1:
  $\mathrm{MST}(V) \subseteq \mathrm{RNG}(V) \subseteq \mathrm{GG}(V) \subseteq \mathrm{DT}(V)$

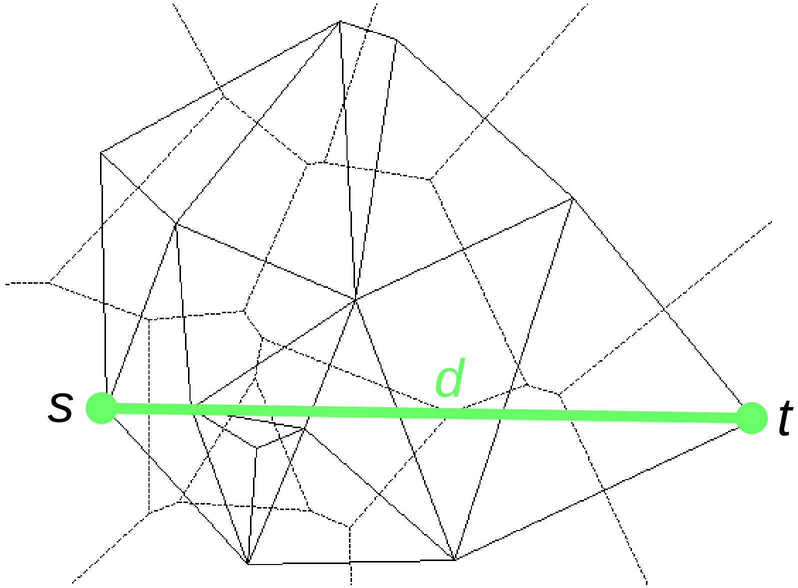
- Corollary:
  Since the MST(V) is connected and the DT(V) is planar, all the planar graphs in Theorem 1 are connected and planar.

- Theorem 2:
  The Gabriel Graph contains the Minimum Energy Path (for any path loss exponent $\alpha \geq 2$)

- Corollary:
  $\mathrm{GG}(V) \cap \mathrm{UDG}(V)$ contains the Minimum Energy Path in UDG(V)

Routing on Delaunay Triangulation?

- Let *d* be the Euclidean distance of source *s* and destination *t*
- Let *c* be the sum of the distances of the links of the shortest path in the Delaunay Triangulation
- It was shown that $c = \Theta(d)$

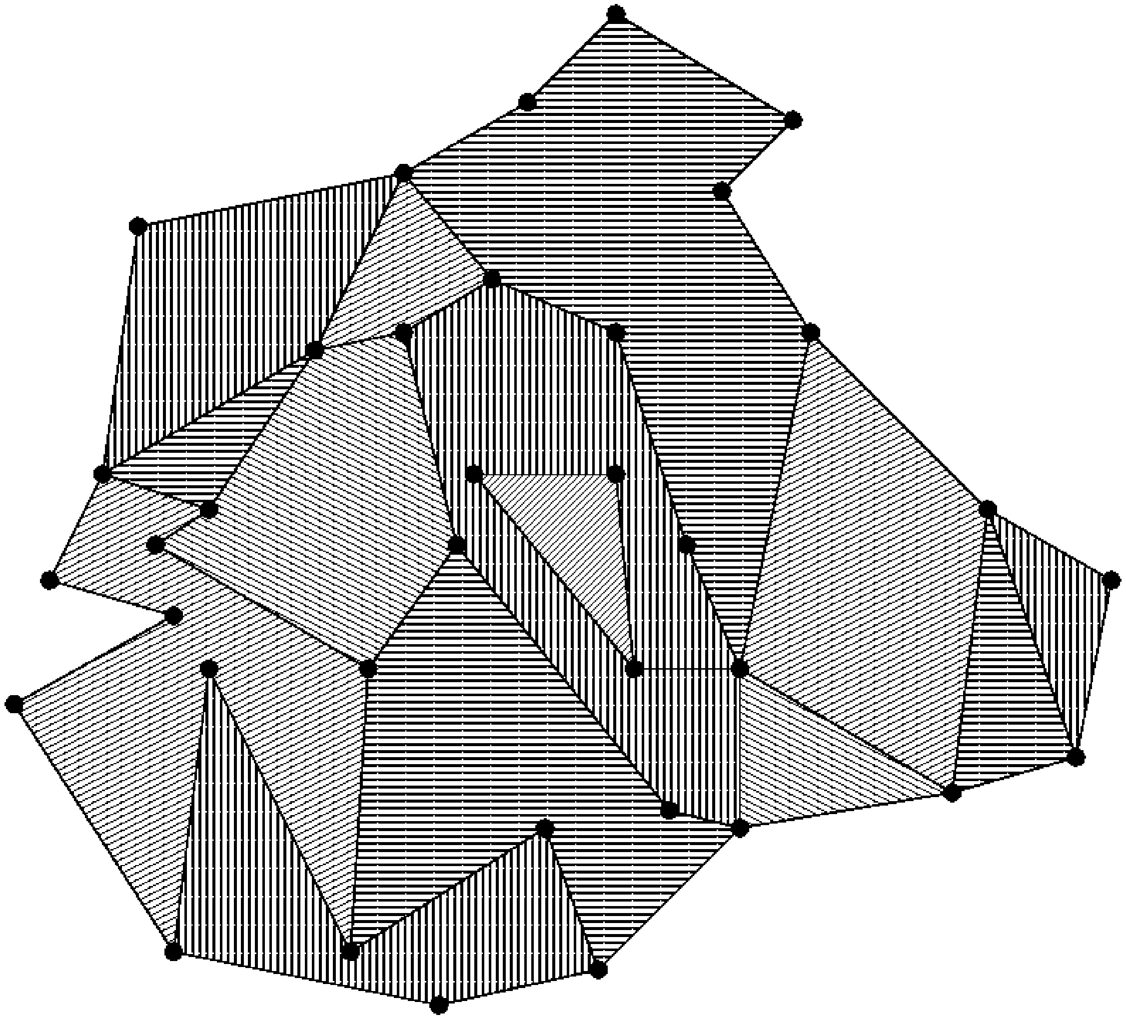


- Three problems:

1) How do we find this best route in the DT? With flooding?!?
2) How do we find the DT at all in a distributed fashion?
3) Worse: The DT contains edges that are not in the UDG, that is, nodes that cannot receive each other are "neighbors" in the DT

Breakthrough idea: route on faces

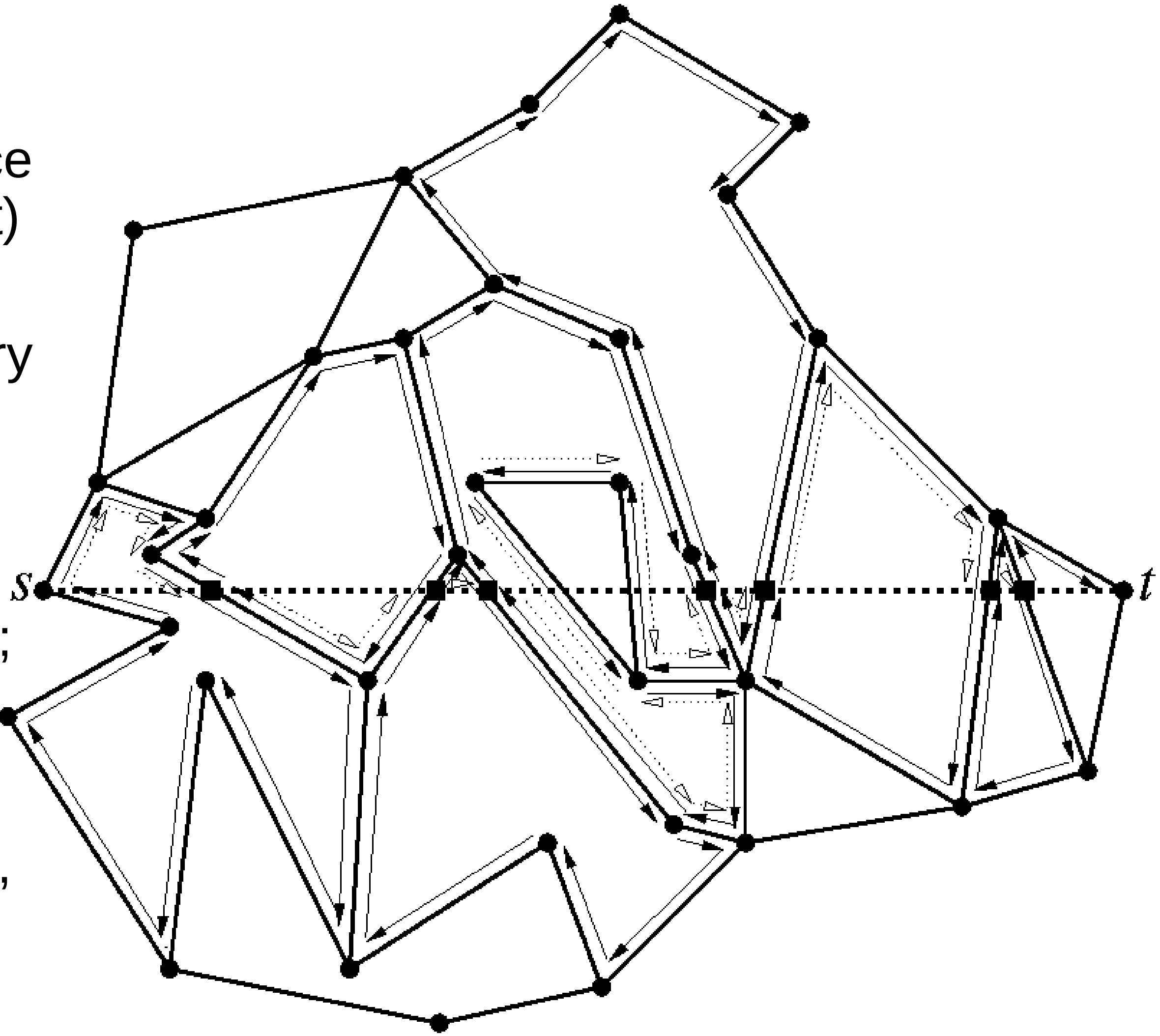
- Remember the faces…
- Idea:
  Route along the boundaries of the faces that lie on the source–destination line



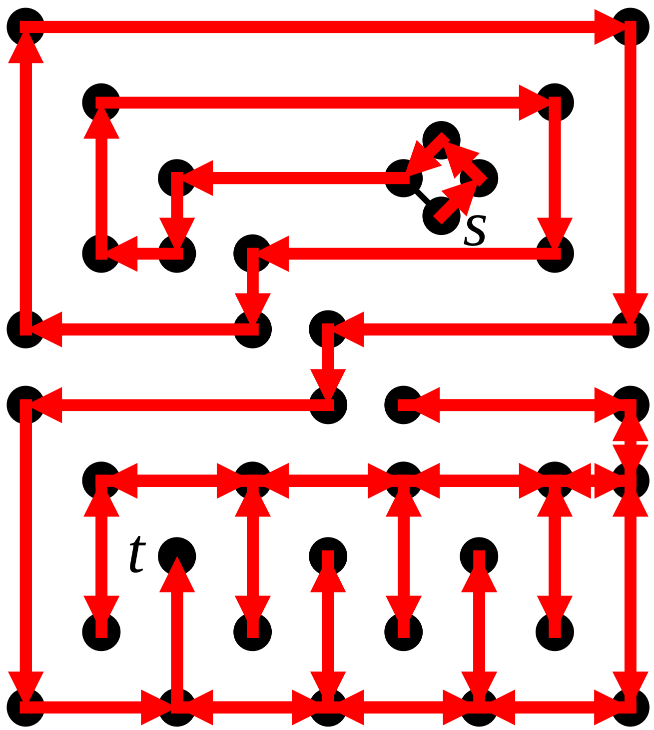
Face Routing

0. Let f be the face incident to the source s, intersected by (s,t)

1. Explore the boundary of f; remember the point p where the boundary intersects with (s,t) which is nearest to t; after traversing the whole boundary, go back to p, switch the face, and repeat 1 until you hit destination t.

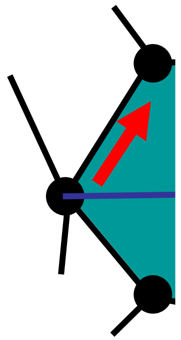


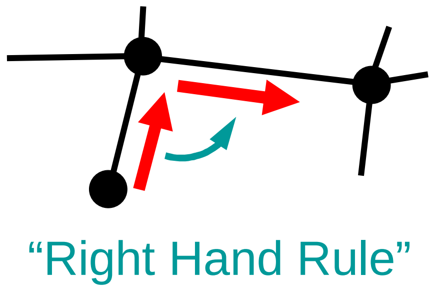
Face Routing Works on *Any* Graph



Face Routing Properties

- All necessary information is stored in the message
  - Source and destination positions
  - Point of transition to next face
- Completely local:
  - Knowledge about direct neighbors‘ positions sufficient
  - Faces are implicit





- Planarity of graph is computed locally (not an assumption)
  - Computation for instance with Gabriel Graph

Face routing is correct

- Theorem: Face routing terminates on any simple planar graph in O(n) steps, where n is the number of nodes in the network

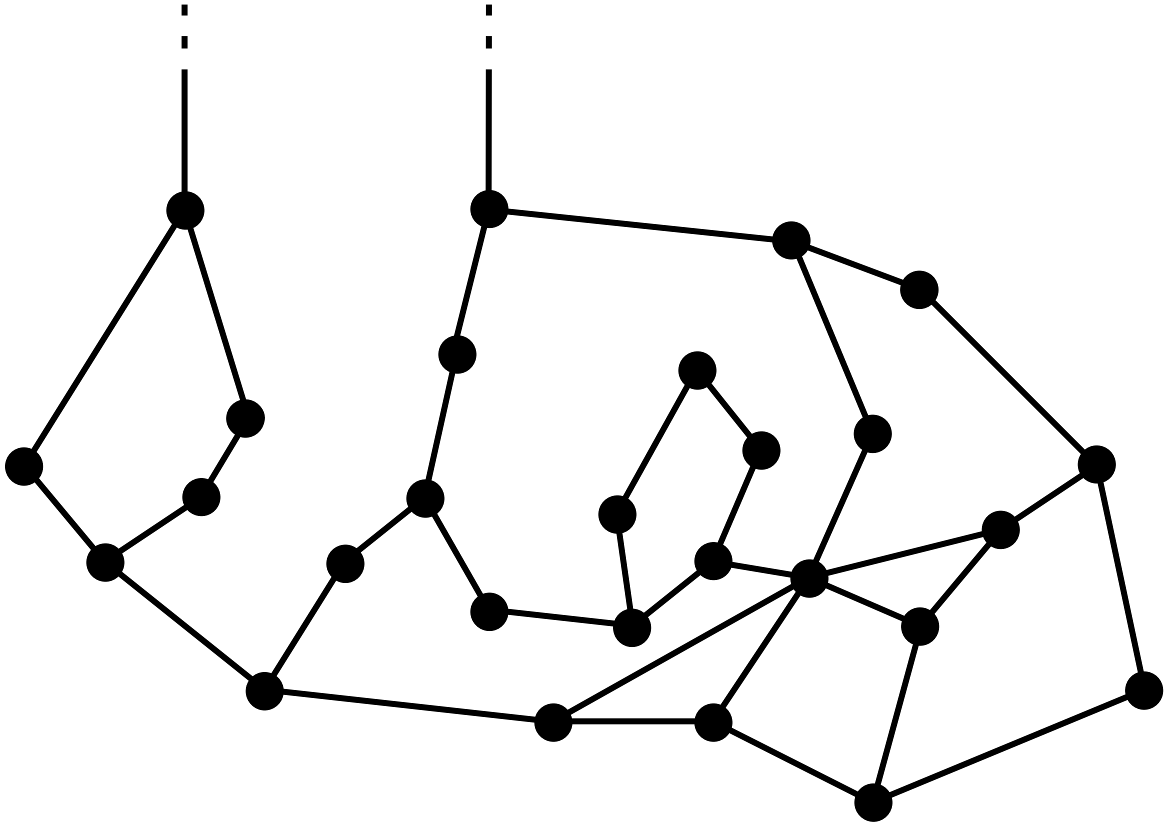
- Proof: A simple planar graph has at most 3n–6 edges. You leave each face at the point that is closest to the destination, that is, you never visit a face twice, because you can order the faces that intersect the source—destination line on the exit point. Each edge is in at most 2 faces. Therefore each edge is visited at most 4 times. The algorithm terminates in O(n) steps.

Is there something better than Face Routing?

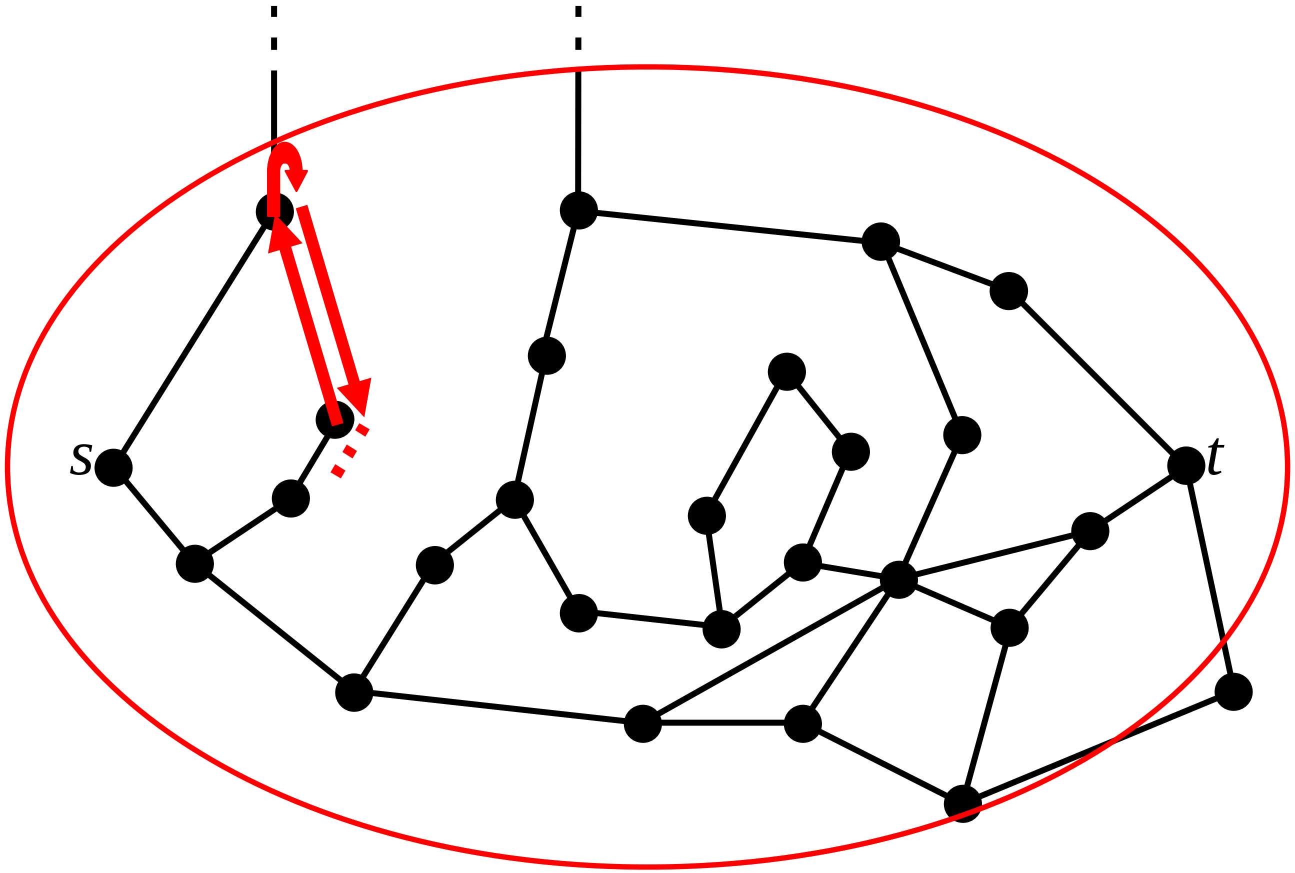
- How to improve face routing? A proposal called "Face Routing 2"
- Idea: Don't search a whole face for the best exit point, but take the first (better) exit point you find. Then you don't have to traverse huge faces that point away from the destination.
- Efficiency: Seems to be practically more efficient than face routing. But the theoretical worst case is worse – $O(n^2)$.
- Problem: if source and destination are very close, we don't want to route through all nodes of the network. Instead we want a routing algorithm where the cost is a function of the cost of the best route in the unit disk graph (and independent of the number of nodes).

Face Routing

- Theorem: Face Routing reaches destination in O(n) steps
- But: Can be very bad compared to the optimal route



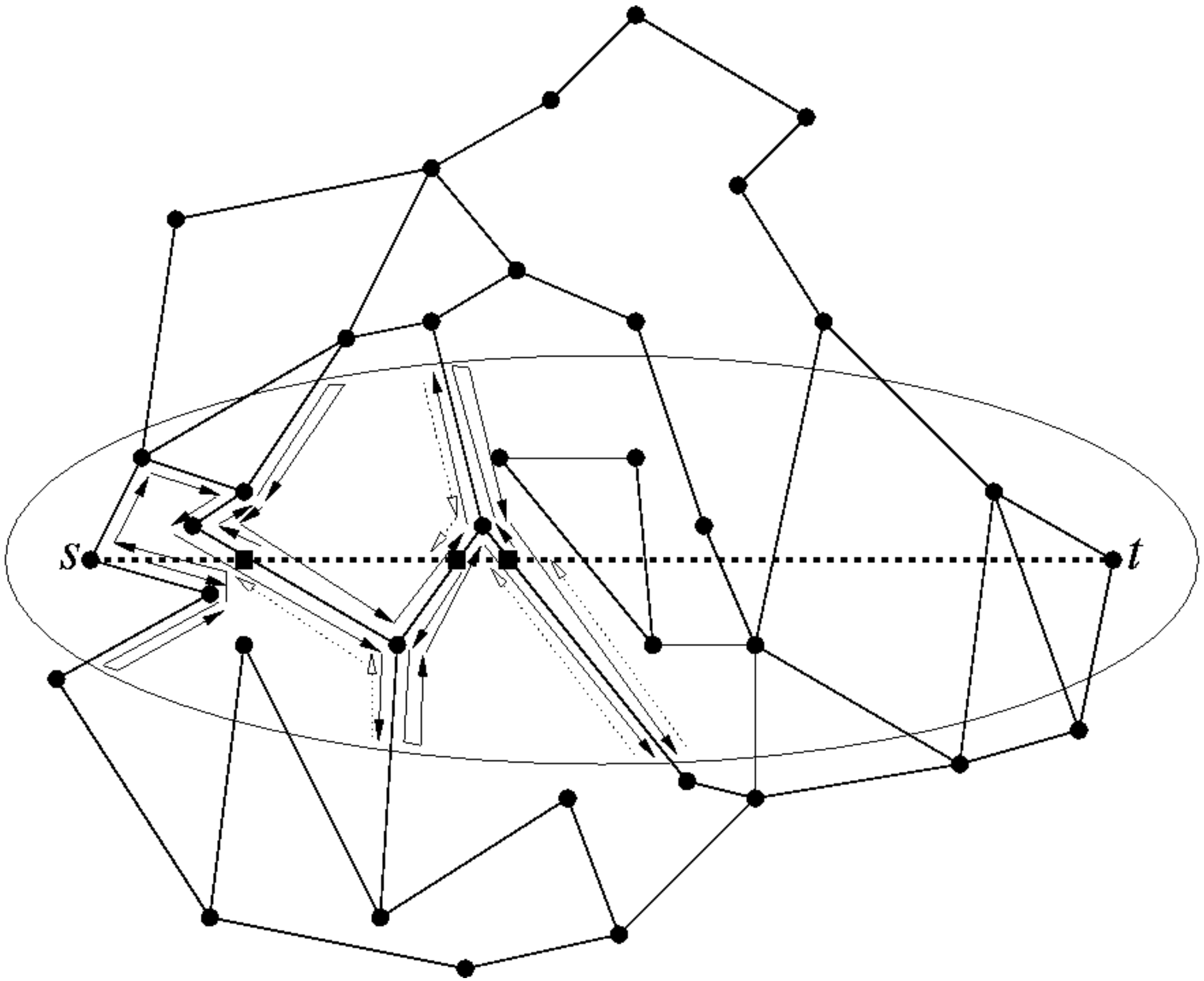
Bounding Searchable Area



Adaptive Face Routing (AFR)

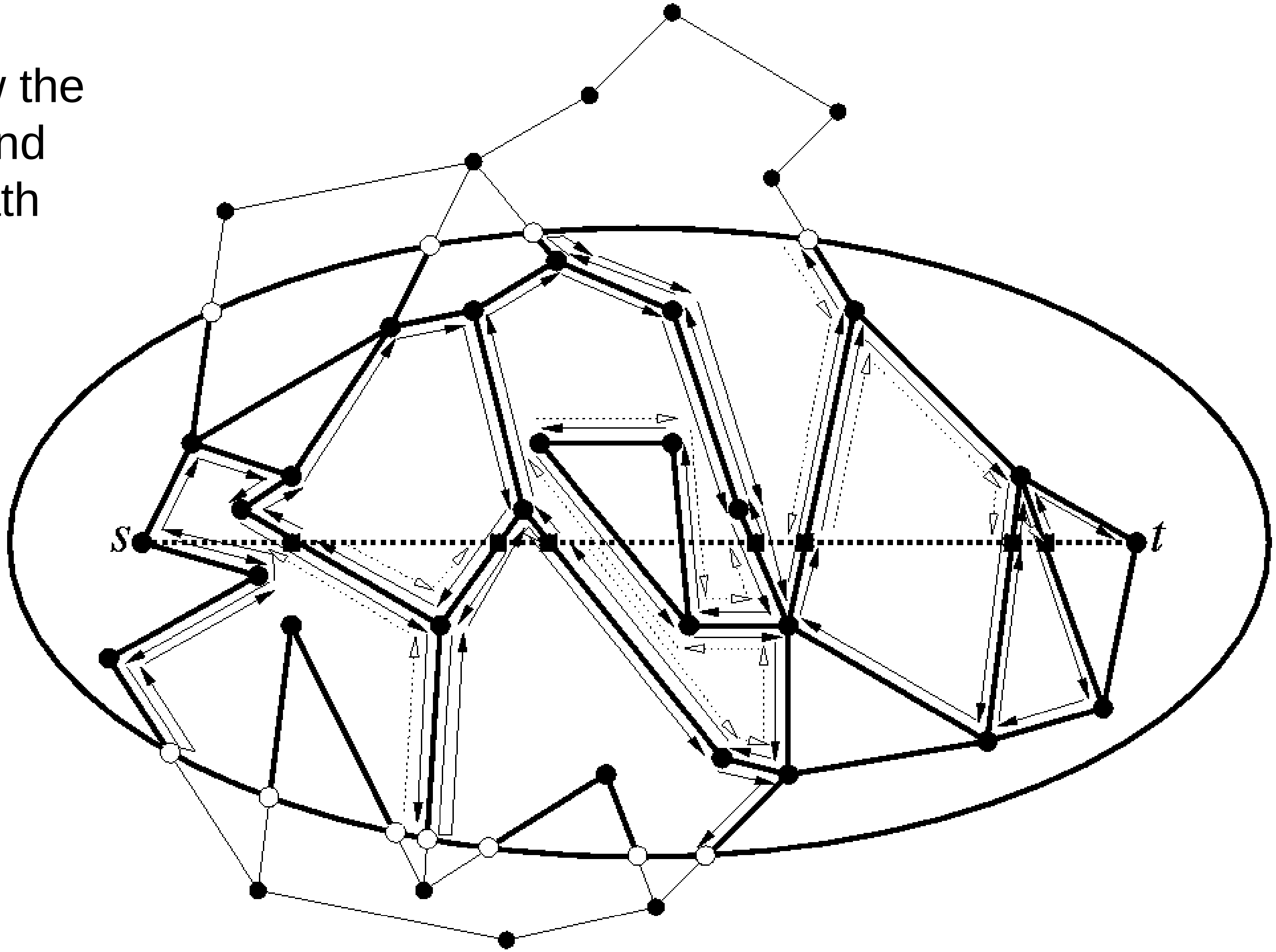
- Idea: Use face routing together with ad hoc routing trick 1!!

- That is, don't route beyond some radius r by branching the planar graph within an ellipse of exponentially growing size.



AFR Example Continued

- We grow the ellipse and find a path



AFR Pseudo-Code

0. Calculate G = GG(V) $\cap$ UDG(V)
   Set c to be twice the Euclidean source—destination distance.

1. Nodes w $\in$ W are nodes where the path s-w-t is larger than c. Do face routing on the graph G, but without visiting nodes in W. (This is like pruning the graph G with an ellipse.) You either reach the destination, or you are stuck at a face (that is, you do not find a better exit point.)

2. If step 1 did not succeed, double c and go back to step 1.

- Note: All the steps can be done completely locally,
  and the nodes need no local storage.

The $\Omega(1)$ Model

- We simplify the model by assuming that nodes are sufficiently far apart; that is, there is a constant d_0 such that all pairs of nodes have at least distance d_0. We call this the $\Omega(1)$ model.

- This simplification is natural because nodes with transmission range 1 (the unit disk graph) will usually not "sit right on top of each other".

- Lemma: In the $\Omega(1)$ model, all natural cost models (such as the Euclidean distance, the energy metric, the link distance, or hybrids of these) are equal up to a constant factor.

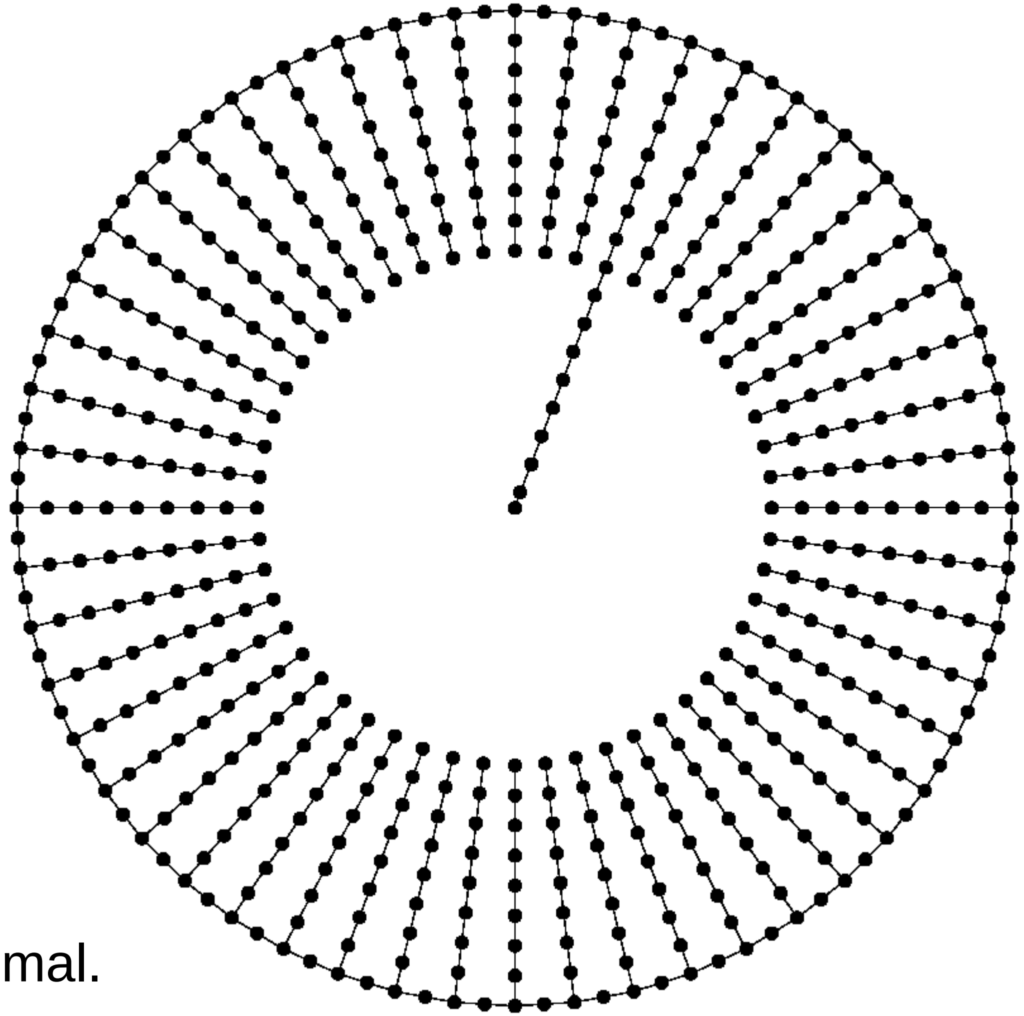
- Remark: The properties we use from the $\Omega(1)$ model can also be established with a backbone graph construction.

Analysis of AFR in the $\Omega(1)$ model

- Lemma 1: In an ellipse of size c there are at most $O(c^2)$ nodes.
- Lemma 2: In an ellipse of size c, face routing terminates in $O(c^2)$ steps, either by finding the destination, or by not finding a new face.
- Lemma 3: Let the optimal source—destination route in the UDG have cost c*. Then this route c* must be in any ellipse of size c* or larger.
- Theorem: AFR terminates with cost $O(c^{*2})$.
- Proof: Summing up all the costs until we have the right ellipse size is bounded by the size of the cost of the right ellipse size.

Lower Bound

- The network on the right constructs a lower bound.
- The destination is the center of the circle, the source any node on the ring.
- Finding the right chain costs $\Omega(c^{*2})$, even for randomized algorithms
- Theorem:
  AFR is asymptotically optimal.



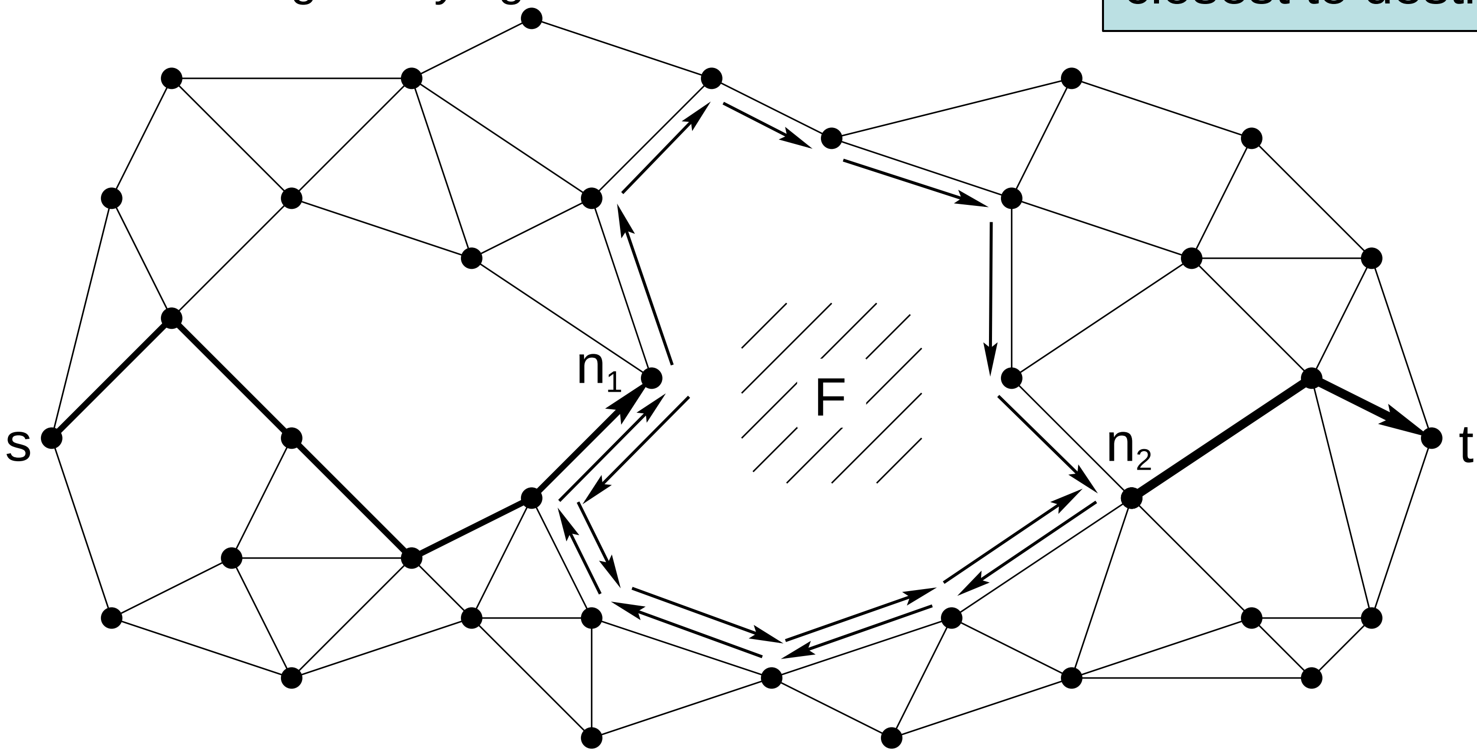
Non-geometric routing algorithms

- In the $\Omega(1)$ model, a standard flooding algorithm enhanced with trick 1 will (for the same reasons) also cost $O(c^{*2})$.
- However, such a flooding algorithm needs O(1) extra storage at each node (a node needs to know whether it has already forwarded a message).
- Therefore, there is a trade-off between O(1) storage at each node or that nodes are location aware, and also location aware about the destination. This is intriguing.

GOAFR – Greedy Other Adaptive Face Routing

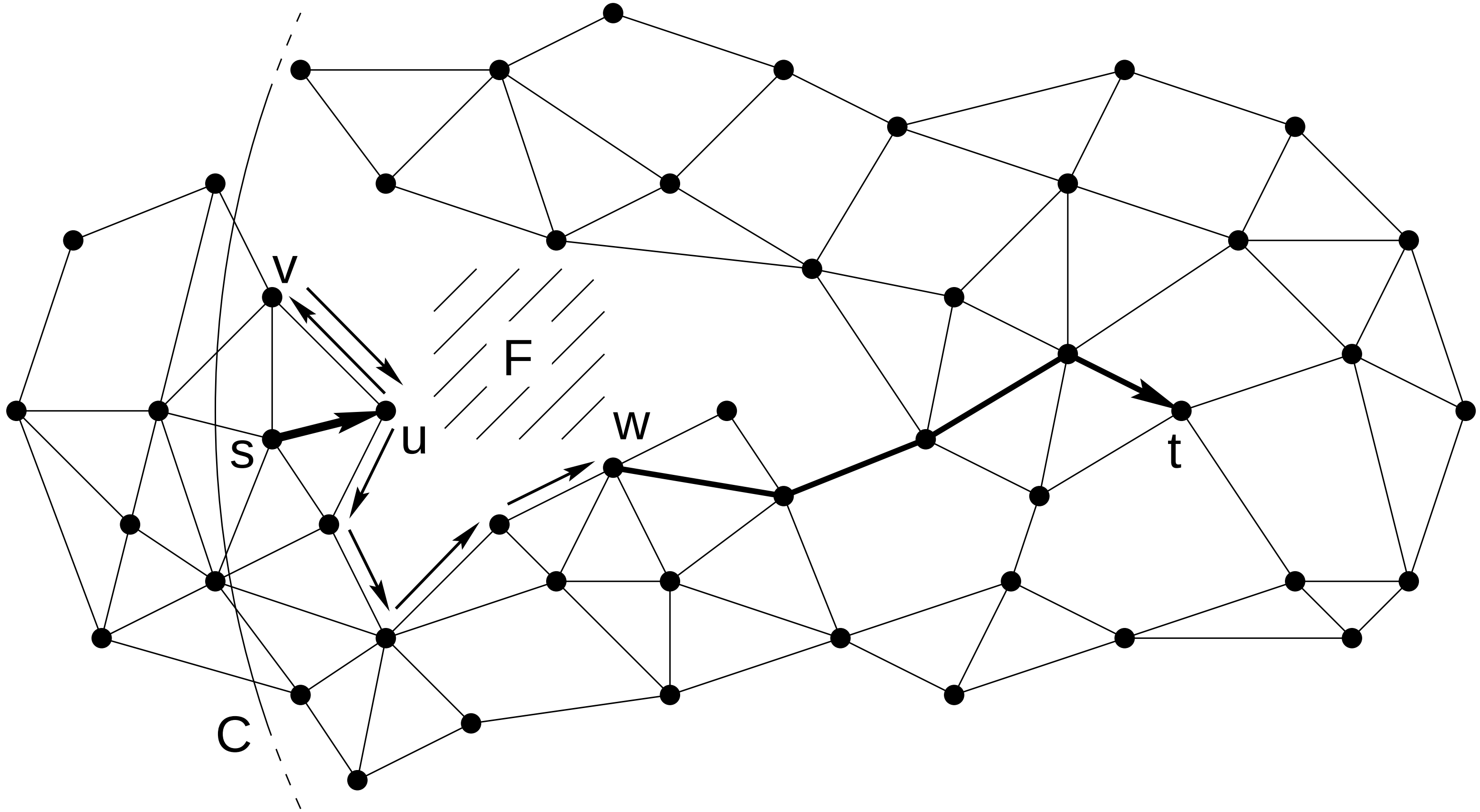
- Back to geometric routing…
- AFR Algorithm is not very efficient (especially in dense graphs)
- Combine Greedy and (Other Adaptive) Face Routing
  - Route greedily as long as possible
  - Circumvent "dead ends" by use of face routing
  - Then route greedily again

Other AFR: In each face proceed to node closest to destination



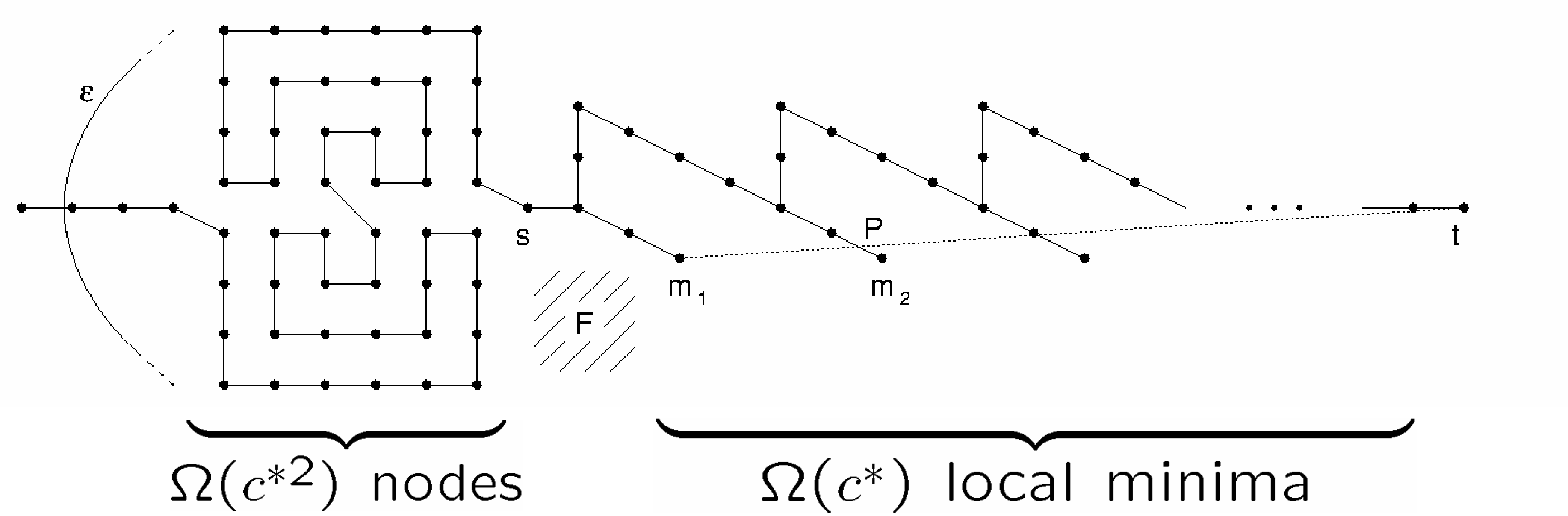
GOAFR+

- GOAFR+ improvements:
  - Early fallback to greedy routing
  - (Circle centered at destination instead of ellipse)



Early Fallback to Greedy Routing?

- We could fall back to greedy routing as soon as we are closer to t than the local minimum
- But:



- "Maze" with $\Omega(c^{*2})$ edges is traversed $\Omega(c^*)$ times $\rightarrow$ $\Omega(c^{*3})$ steps

GOAFR – Greedy Other Adaptive Face Routing

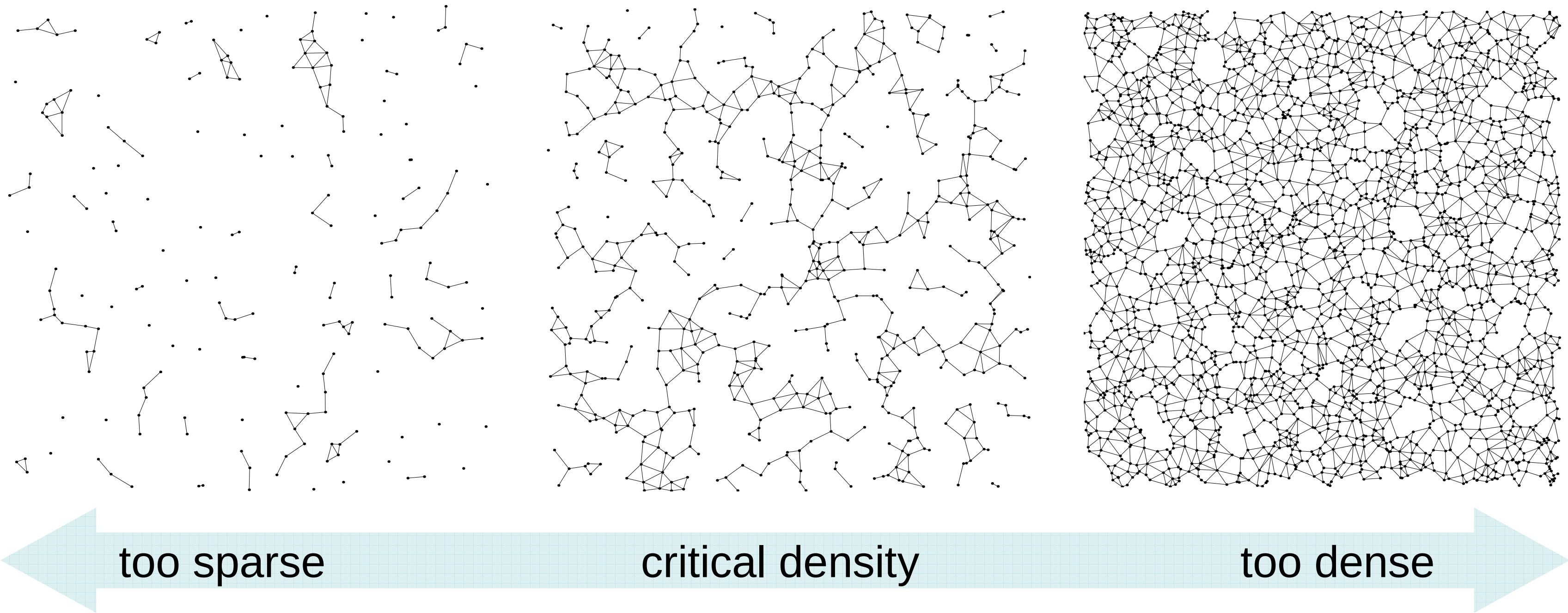
- Early fallback to greedy routing:
  - Use counters p and q. Let u be the node where the exploration of the current face F started
    - p counts the nodes closer to t than u
    - q counts the nodes *not* closer to t than u
  - Fall back to greedy routing as soon as $p > \sigma \cdot q$ (constant $\sigma > 0$)

> Theorem: GOAFR is still asymptotically worst-case optimal…
> …*and* it is efficient in practice, in the average-case.

- What does "practice" mean?
  - Usually nodes placed uniformly at random

Average Case

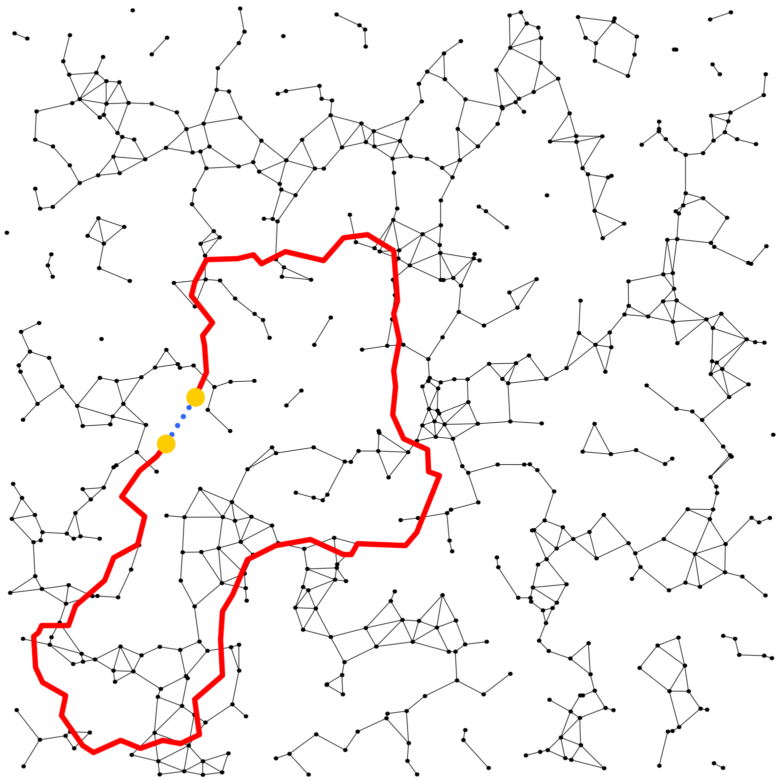
- Not interesting when graph not dense enough
- Not interesting when graph is too dense
- Critical density range ("percolation")
  - Shortest path is significantly longer than Euclidean distance



Critical Density: Shortest Path vs. Euclidean Distance

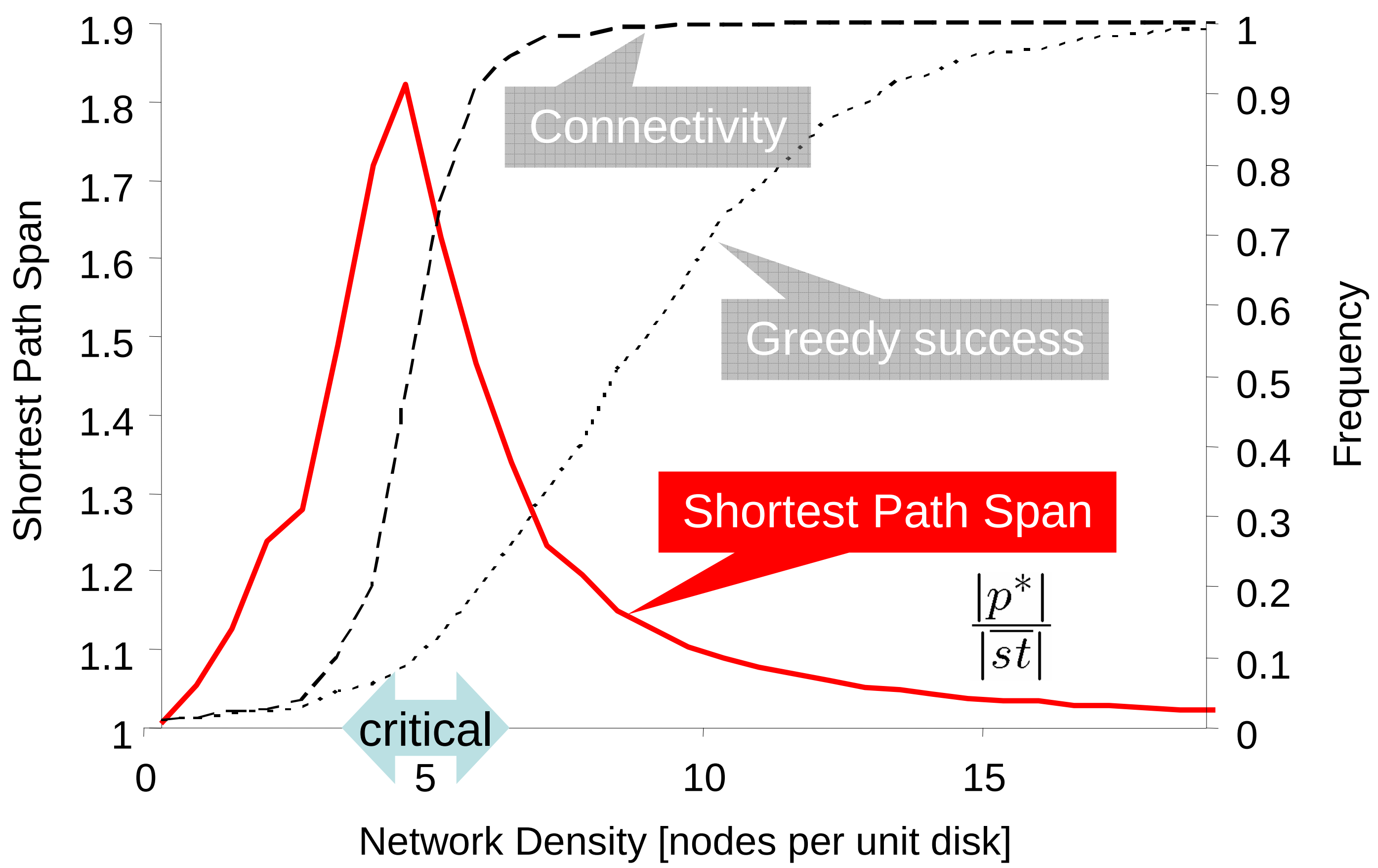
- Shortest path is significantly longer than Euclidean distance

$$\frac{|p^*|}{|\overline{st}|}$$

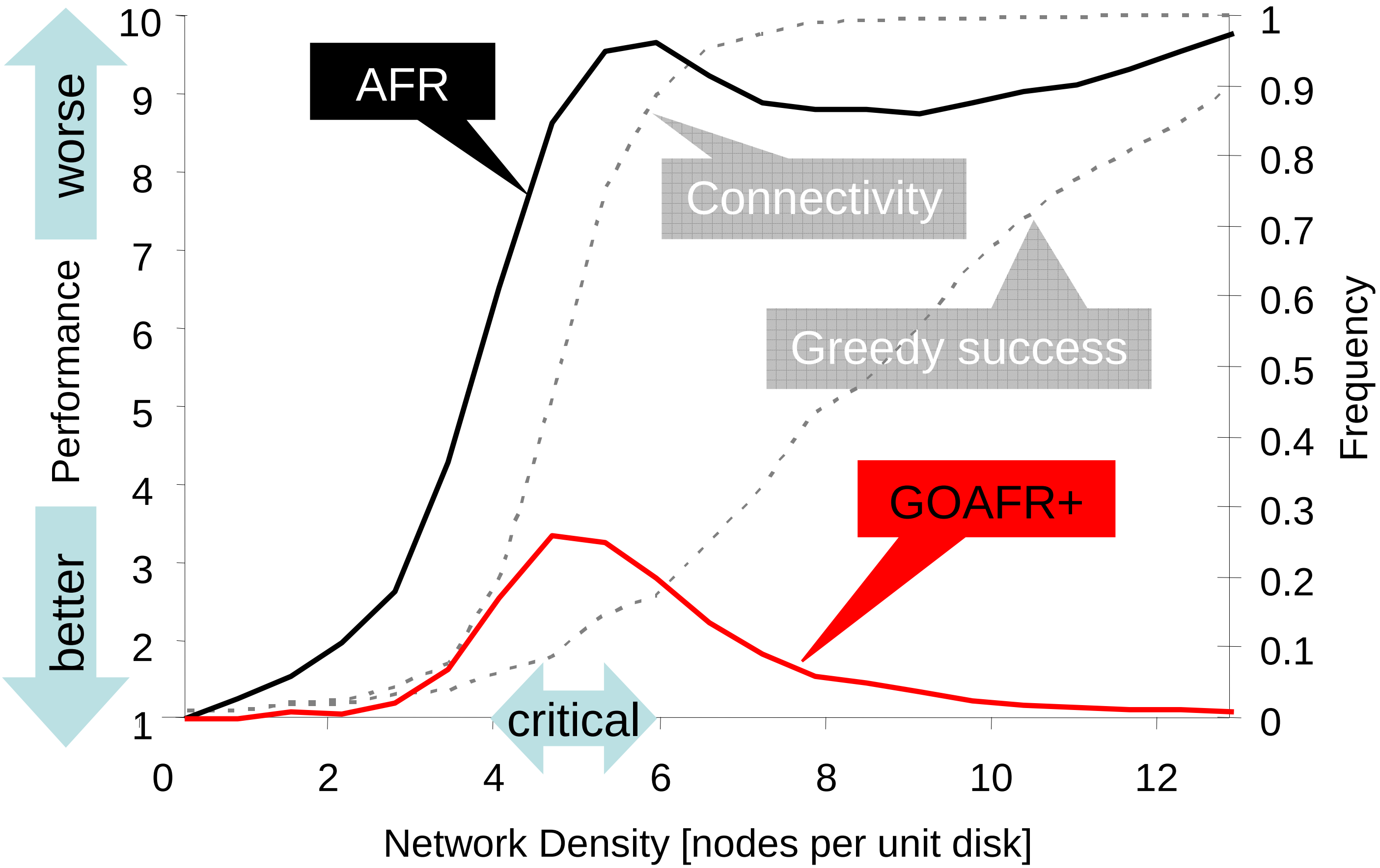


- Critical density range mandatory for the simulation of any routing algorithm (not only geographic)

Randomly Generated Graphs: Critical Density Range



Simulation on Randomly Generated Graphs



A Word on Performance

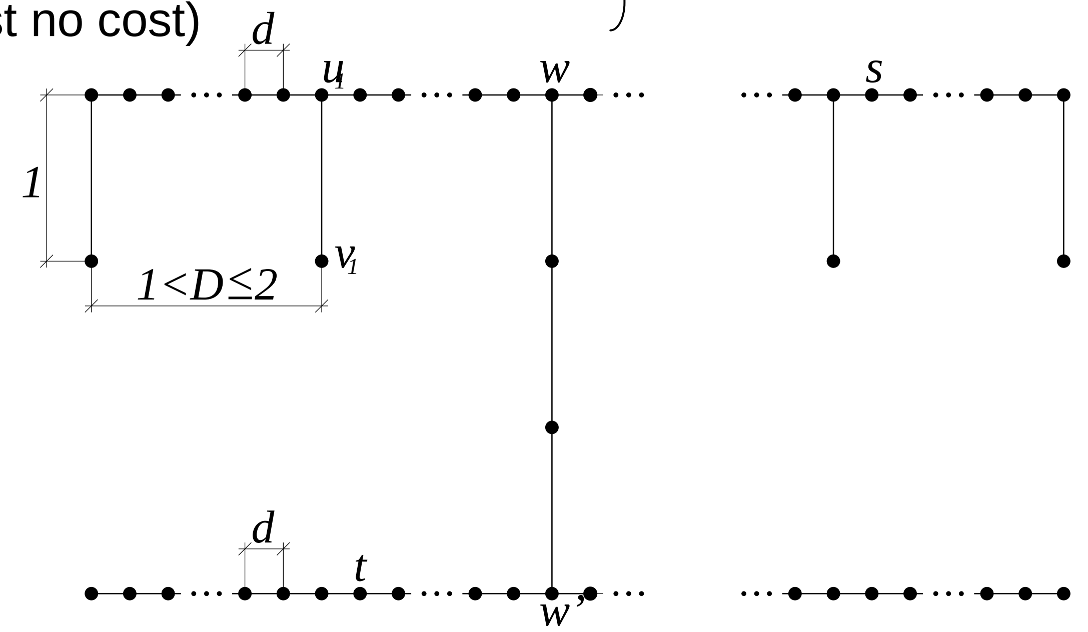
- What does a performance of 3.3 in the critical density range mean?
- If an optimal path (found by Dijkstra) has cost c, then GOAFR+ finds the destination in 3.3·c steps.
- It does *not* mean that the *path* found is 3.3 times as long as the optimal path! The path found can be much smaller…
- Remarks about cost metrics
  - In this lecture "cost" c = c hops
  - There are other results, for instance on distance/energy/hybrid metrics
  - In particular: With energy metric there is no competitive geometric routing algorithm

Energy Metric Lower Bound

Example graph: k "stalks", of which only one leads to t

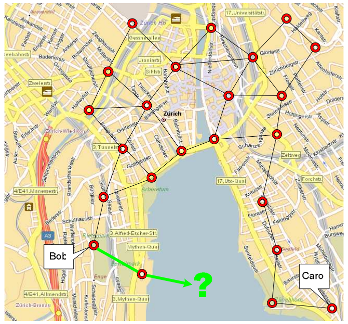
- any deterministic (randomized) geometric routing algorithm A has to visit all k (at least k/2) "stalks"
- optimal path has constant cost c^* (covering a constant distance at almost no cost)

$$\lim_{k \to \infty} \frac{c(A)}{c^*} = \infty$$

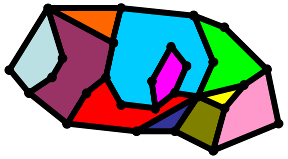


→ With energy metric there is no competitive geometric routing algorithm

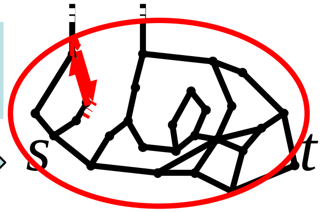
GOAFR: Summary



Face Routing

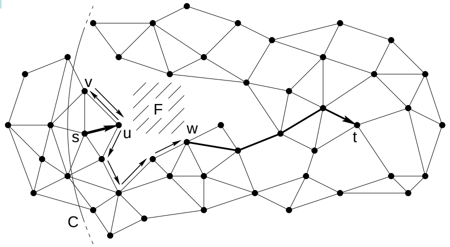


Adaptive Face Routing



Greedy Routing

GOAFR+



Average-case efficiency

Worst-case optimality

"Practice"

"Theory"

Routing with and without position information

- Without position information:
  - Flooding

    → does not scale
  - Distance Vector Routing

    → does not scale
  - Source Routing
    - increased per-packet overhead
    - no theoretical results, only simulation

- With position information:
  - Greedy Routing

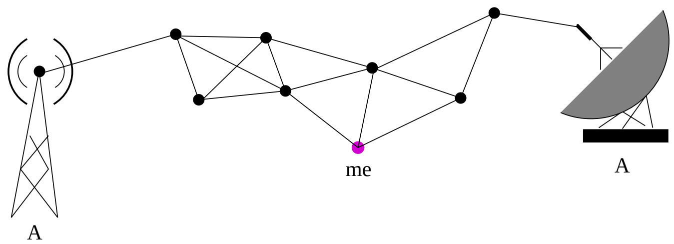
    → may fail: message may get stuck in a "dead end"
  - Geometric Routing

    → It is assumed that each node knows its position

Obtaining Position Information

- Attach GPS to each sensor node
  - Often undesirable or impossible
  - GPS receivers clumsy, expensive, and energy-inefficient

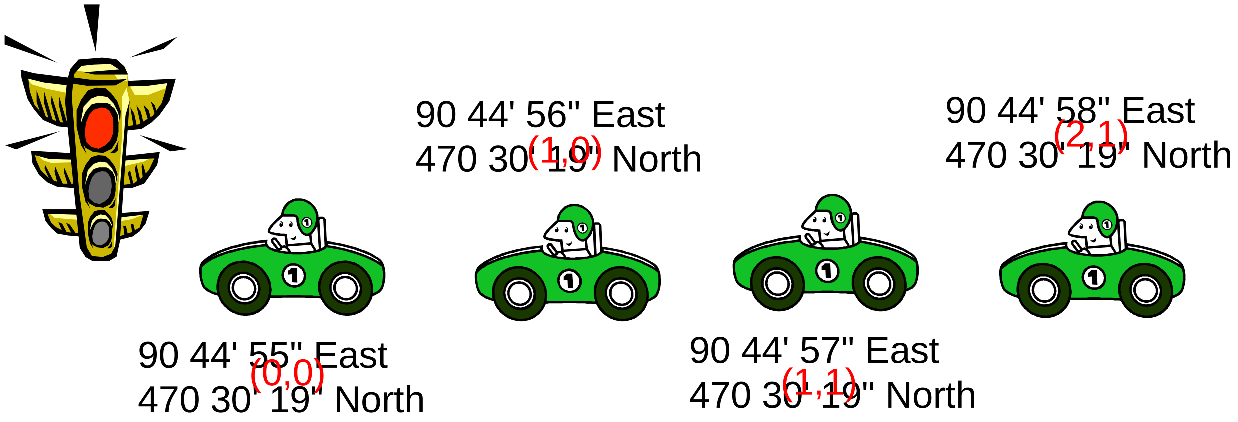
- Equip only a few designated nodes with a GPS
  - Anchor (landmark) nodes have GPS
  - Non-anchors derive their position through communication (e.g., count number of hops to different anchors)



Anchor density determines quality of solution

What about no GPS at all?

- In absence of GPS-equipped anchors...
  → ...nodes are clueless about real coordinates.
- For many applications, real coordinates are not necessary
  → Virtual coordinates are sufficient



real coordinates

virtual coordinates

What are „good“ virtual coordinates?

- Given the connectivity information for each node and knowing the underlying graph is a UDG find virtual coordinates in the plane such that all connectivity requirements are fulfilled, i.e. find a realization (embedding) of a UDG:
  - each edge has length at most 1
  - between non-neighbored nodes the distance is more than 1

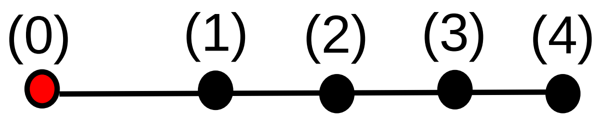
- Finding a realization of a UDG from connectivity information only is NP-hard...
  - [Breu, Kirkpatrick, Comp.Geom.Theory 1998]
- ...and also hard to approximate
  - [Kuhn, Moscibroda, Wattenhofer, DIALM 2004]

Geometric Routing without Geometry

- For many applications, like routing, finding a realization of a UDG is not mandatory
- Virtual coordinates merely as infrastructure for geometric routing

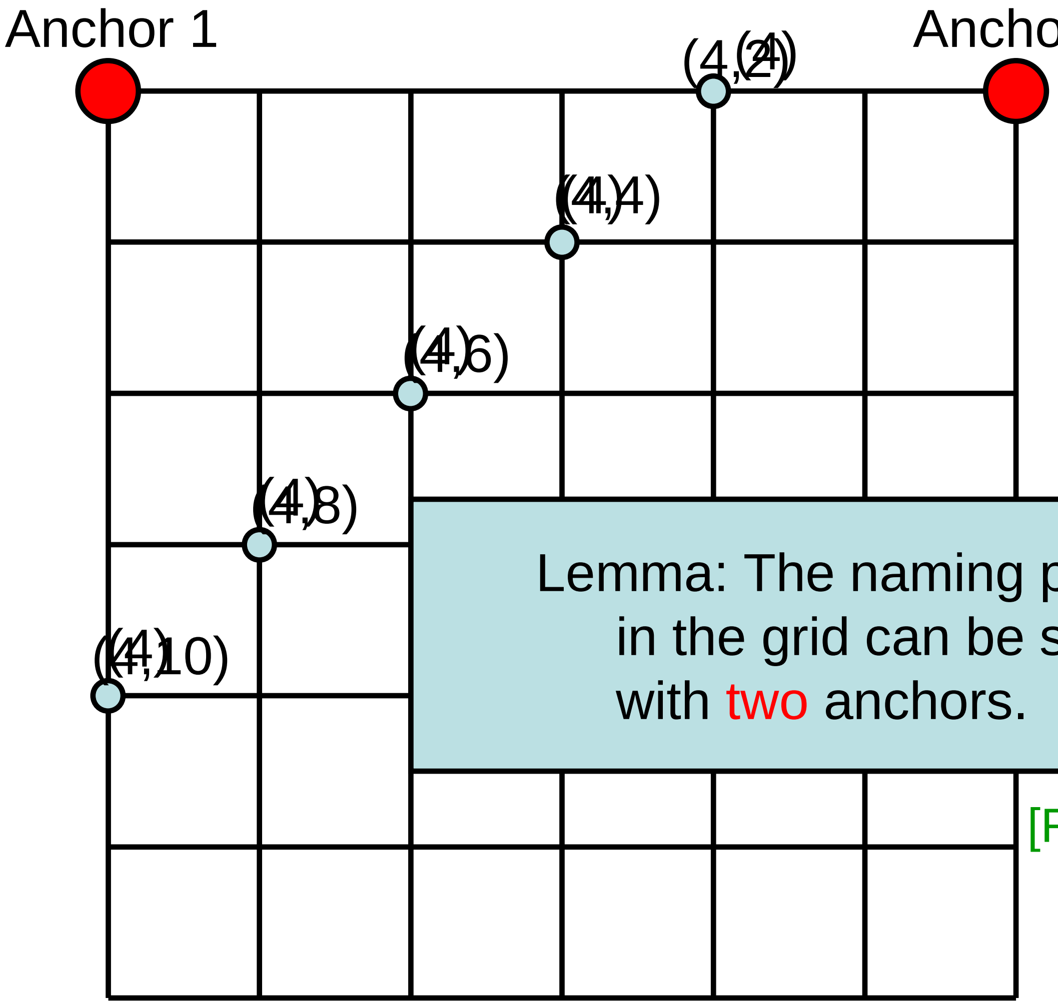
→ Pseudo geometric coordinates:

  - Select some nodes as anchors: $a_1, a_2, ..., a_k$
  - Coordinate of each node *u* is its hop-distance to all anchors: $(d(u,a_1), d(u,a_2), ..., d(u,a_k))$



- Requirements:
  - each node uniquely identified: Naming Problem
  - routing based on (pseudo geometric) coordinates possible: Routing Problem

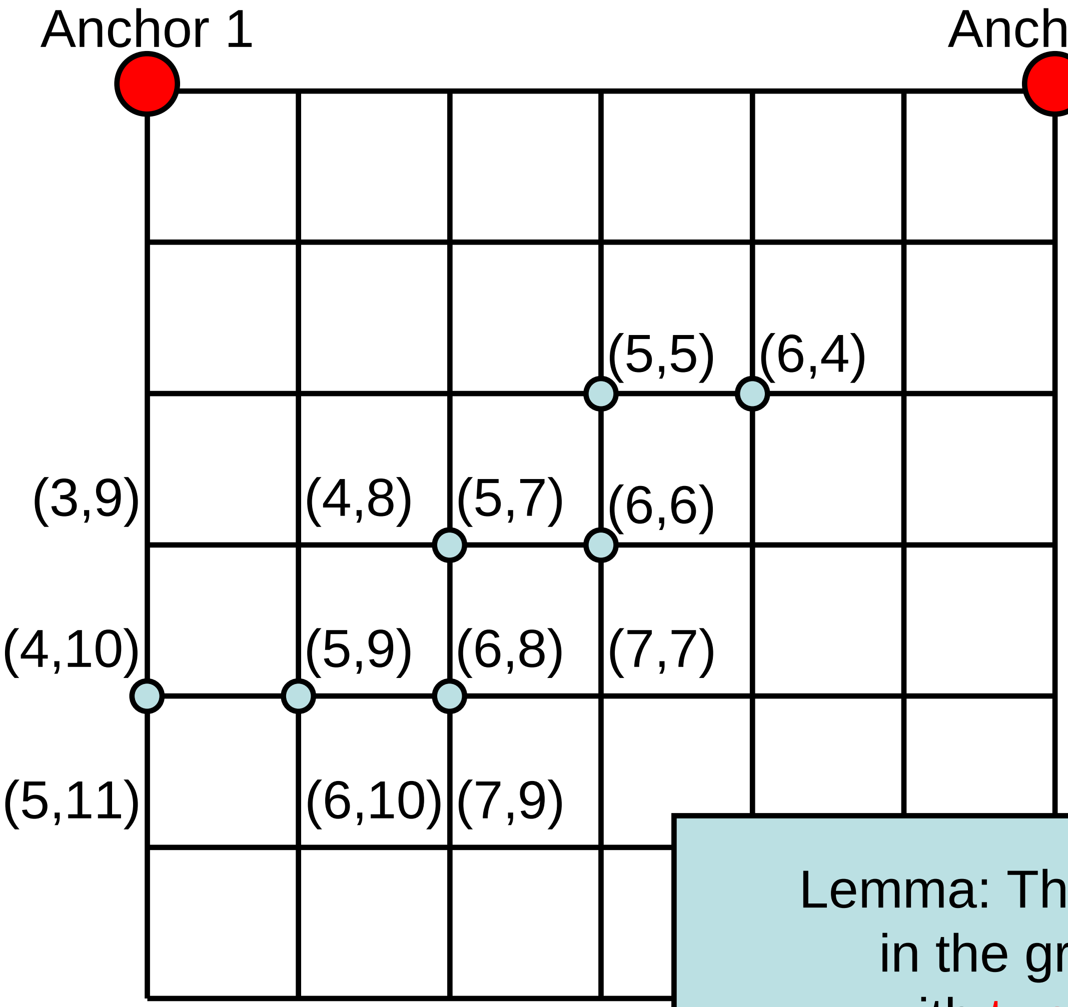
Pseudo-geometric routing in the grid: Naming



Lemma: The naming problem in the grid can be solved with two anchors.

[R.A. Melter and I. Tomescu, Comput. Vision, Graphics. Image Process., 1984]: landmarks in graphs

Pseudo-geometric routing in the grid: Routing

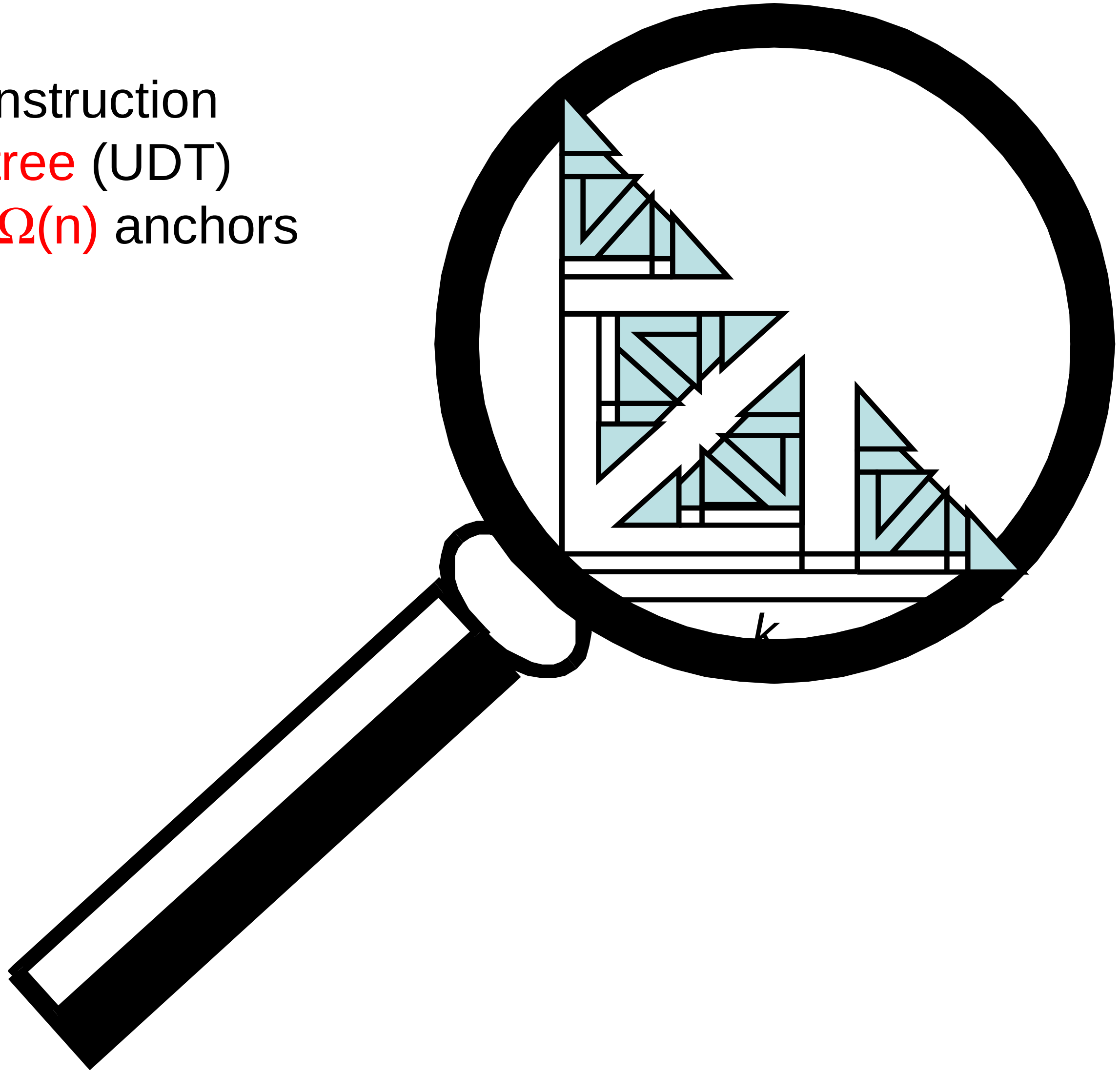


Rule: pass message to neighbor which is closest to destination

Lemma: The routing problem in the grid can be solved with two anchors.

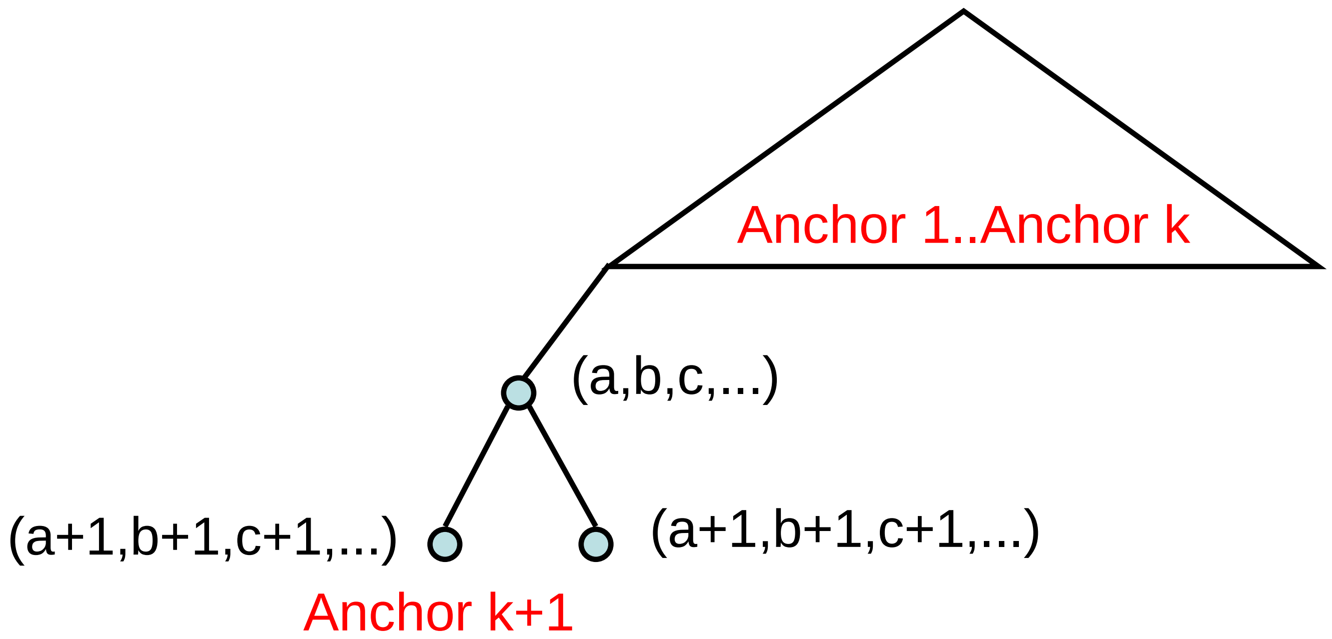
Problem: UDG is usually not a grid

- Recursive construction of a unit dist tree (UDT) which needs $\Omega(n)$ anchors



Pseudo-geometric routing in the UDT: Naming

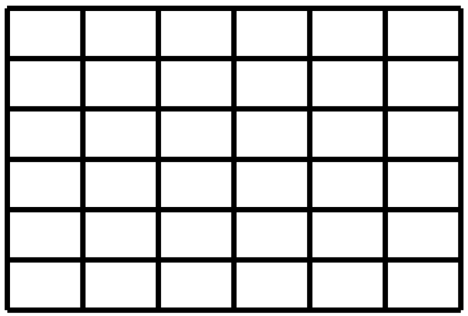
- Leaf-siblings can only be distinguished if one of them is an anchor:

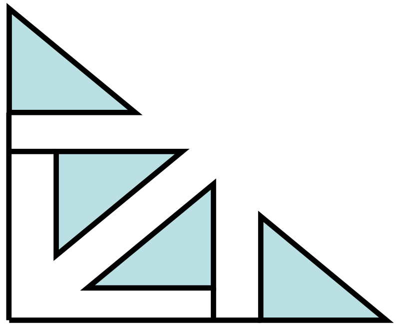


Lemma: in a unit disk tree with *n* nodes there are up to *Θ(n)* leaf-siblings. That is, we need to *Θ(n)* anchors.

Pseudo-geometric routing in the ad hoc networks

- Naming and routing in grid quite good, in previous UDT example very bad
- Real-world ad hoc networks are very probable neither perfect grids nor naughty unit disk trees





Truth is somewhere in between...