Geo-Routing

Thanks to Stefan Schmid for slides

Overview

- Classic routing overview
- Geo-routing
- Greedy geo-routing

- Euclidean and Planar graphs
- Face Routing
- Greedy and Face Routing

Shortest path

- An important issue is: how well do such algorithms perform when the topology changes? No real network is static!

- Let us examine ***distance vector routing*** that is adaptation of the shortest path algorithm

Distance Vector Routing

- Distance vector routing uses the basic idea of shortest path routing, but handles topology changes.
- The routing table is an array of tuples <destination, nexthop, distance>.
- To send a packet to a given destination, it is forwarded to the process in the corresponding nexthop field of the tuple.
- When a node j or a link crashes some neighbor of it detects the failure and sets the corresponding distance to ∞.
- When a new node joins the network, or an existing node is repaired, the neighbor detecting it sets the corresponding distance to 1.
- Routing table is eventually recomputed.
- Unfortunately, depending on when a failure is detected, and when the advertisements are sent out, the routing table may not stabilize soon.

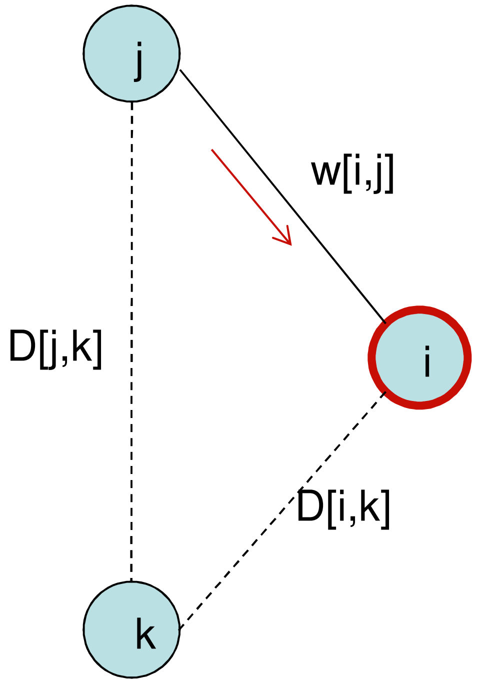
Distance Vector Routing

Distance Vector D for each node **i** contains **N** elements **D[i,0], D[i,1], D[i,2]** … **D[i, N-1]**. **D[i,j]** denotes the distance from node **i** to node **j.** ∀i, D[i,i] =0, and initially ∀i,j: i≠j, D[i,j] = ∞.

- Each node **j** periodically sends its distance vector to its immediate neighbors.

- Every neighbor **i** of **j**, after receiving the broadcasts from its neighbors, updates its distance vector as follows:

$$\forall\, k \neq i: D[i,k] = \min_k(w[i,j] + D[j,k])$$



Suggested Reading:
Routing Information Protocol (RIP),
Interior Gateway Routing Protocol (IGRP).

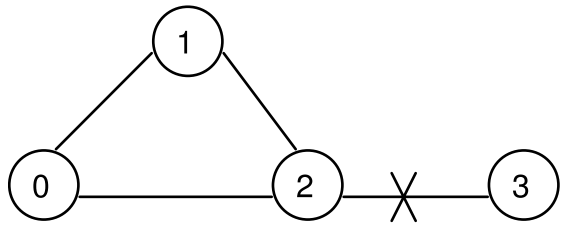
What if the topology changes?

Assume that each edge has weight = 1. Currently,

Node 1: d(1,0) = 1, d(1, 2) = 1, d(1,3) = 2

Node 2: d(2,0) = 1, d(2,1) =1, d(2,3) = 1

Node 1: d(3,0) = 2, d(3,1) = 2, d(3,2) = 1



Observe what can happen when the link (2,3) fails.

Counting to infinity

Observe what can happen when the link (2,3) fails.

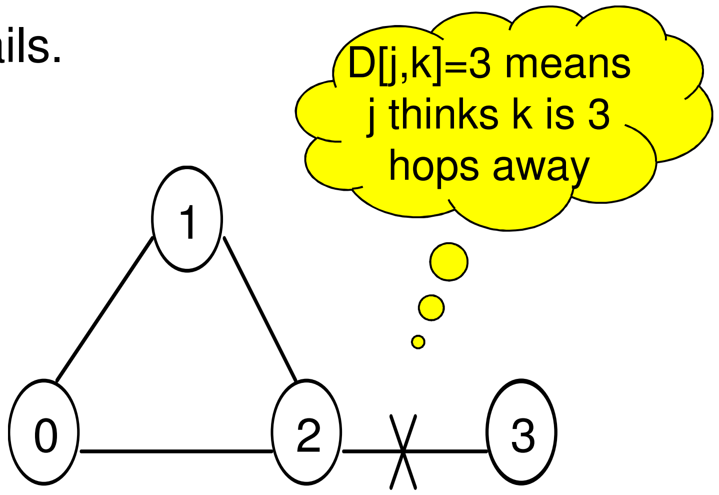
Node 1 thinks d(1,3) = 2 (old value)

Node 2 thinks d(2,3) = d(1,3) +1 = 3

Node 1 thinks d(1,3) = d(2,3) +1 = 4

…

and so on. So it will take forever for the distances to stabilize.



$\forall$ k≠ i: D[i,k] = min$_k$(w[i,j] + D[j,k])

- A partial remedy is the **split horizon** method that will prevent node 1 from sending the advertisement about d(1,3) to 2 since its first hop (to 3) is node 2.

Suitable for smaller networks. Larger volume of data is disseminated, but to its immediate neighbors only. Poor convergence property

Link State Routing

- This is an alternative method of shortest path routing
- In comparison with distance vector routing, link-state routing protocol converges faster.
- Each node **i** periodically broadcasts the weights of all edges **(i,j)** incident on it (this is the *link state*) to all its neighbors. The mechanism for dissemination is flooding.
- Link state broadcasts are sent out reliable flooding, which guarantees that the broadcasts reach every node.
- This helps each node eventually compute the topology of the network, and independently determine the shortest path to any destination node using some standard **sequential graph algorithm like Dijkstra's.**
- When failures are not taken into consideration, the correctness follows trivially. The total number of LSPs circulating in the network for every change in the link state is |**E**|**.**

Smaller volume data disseminated over the entire network
Used in *Open Shortest Path First* (OSPF) of *Internet Protocol* (IP)

Link State Routing contd..

- The failure (or temporary unavailability) of links and nodes can make the algorithm more complicated.
- When a node i crashes, the link-state packet s (LSPs) stored in it are lost — so it has to reconstruct the topology from the newer packets**.**
- New link states replace the old ones in case of links and nodes failure and repair taken place.
- The links may not be FIFO, so to distinguish between the old and the new link states each link state contains a **sequence number seq**.
- Each link state packet has a **seq** that reflects the order in which the packets were generated. While sending a LSP, a node increments *its* **seq** by 1.
- Each node records the largest **seq** received from *every other node*. Packets with higher **seq** are more recent, and used for updates. Packets with lower **seq** are considered old, and **discarded**.

Link State Routing: clarification

When a node crashes, all packets stored in it may be lost.

After it is repaired, new packets are sent with **seq = 0**.

So these new packets may be discarded in favor of the old packets!

Problem resolved using **time-to-live** (TTL)

Time-To-Live (TTL)

Each LSP contains a TTL field, which is an estimate of the time after which a packet should be considered **stale** (out of date), and discarded. Every node decrements the TTL field of all its LSPs at a steady rate.1 Furthermore, every time a node forwards a stored LSP, it decrements its TTL.

When the TTL of a packet becomes 0, the packet is discarded.

Of course transient failures can corrupt ***seq*** in an unpredictable manner and challenge the protocol.

Corrupt LSP entries are eventually flushed out using the TTL field.

Suggested reading: **Dynamic Routing Protocols by Jeff Doyle**, Sample Chapter is provided courtesy of Cisco Press, Nov 16, 2001. See:
http://www.ciscopress.com/articles/article.asp?p=24090&seqNum=4

Discussion of Classic Routing Protocols

- Proactive Routing Protocols
- Both link-state and distance vector are "proactive," that is, routes are established and updated even if they are never needed.
- If there is almost no mobility, proactive algorithms are superior because they never have to exchange information and find optimal routes easily.

- Reactive Routing Protocols
- Flooding is "reactive," but does not scale
- If mobility is high and data transmission rare, reactive algorithms are superior; in the extreme case of almost no data and very much mobility the simple flooding protocol might be a good choice.

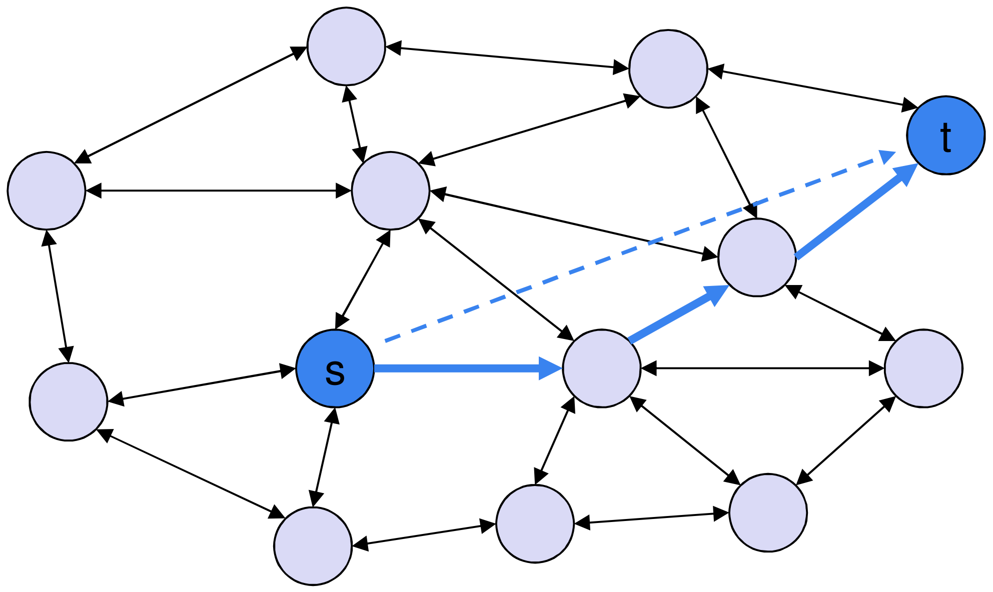
There is *no* "optimal" routing protocol; the choice of the routing protocol depends on the circumstances. Of particular importance is the mobility/data ratio.

Routing in Ad-Hoc Networks

- Reliability
  - Nodes in an ad-hoc network are not 100% reliable
  - Algorithms need to find alternate routes when nodes are failing
- Mobile Ad-Hoc Network (MANET)
  - It is often assumed that the nodes are mobile (“Car2Car”)
- Q: How good are these routing algorithms?!? Any hard results?
- A: Almost none! Method-of-choice is simulation…

Geometric (geographic, directional, position-based) routing

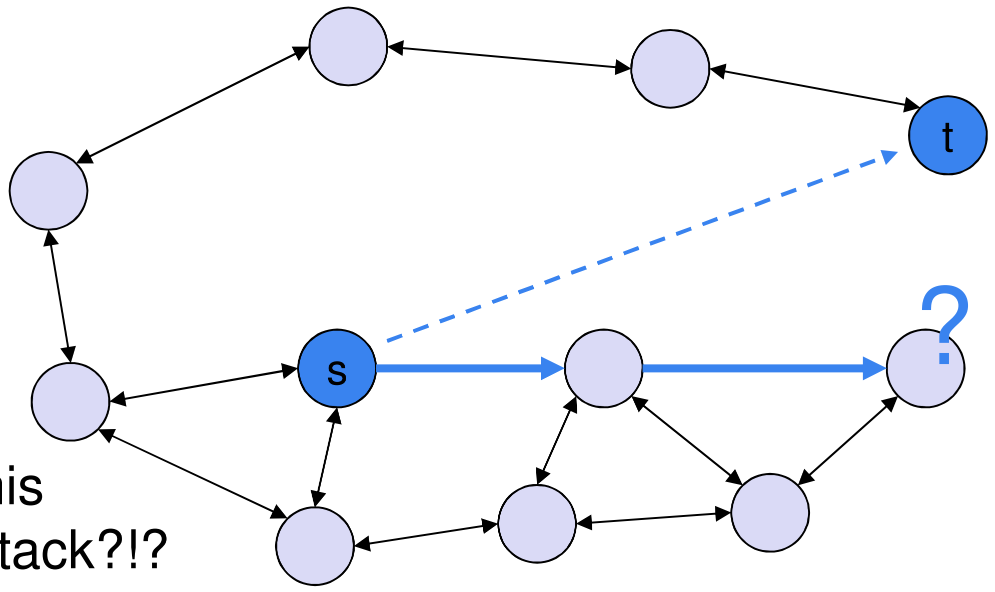
- …even with all the tricks there will be flooding every now and then.
- In this part we will assume that the nodes are location aware (they have GPS, or an ad-hoc way to figure out their coordinates), and that we know where the destination is.
- Then we simply route towards the destination



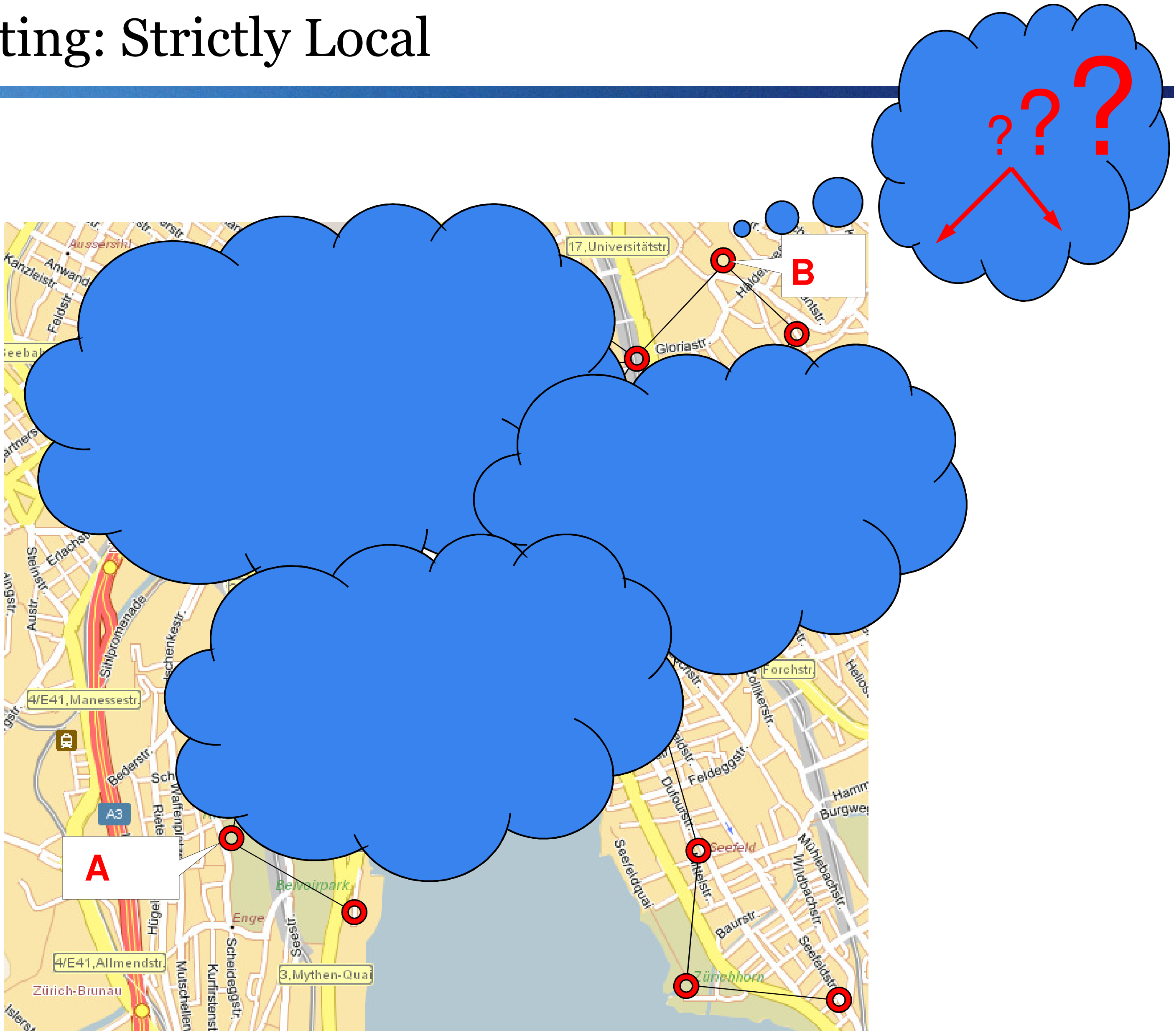
Geometric routing

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

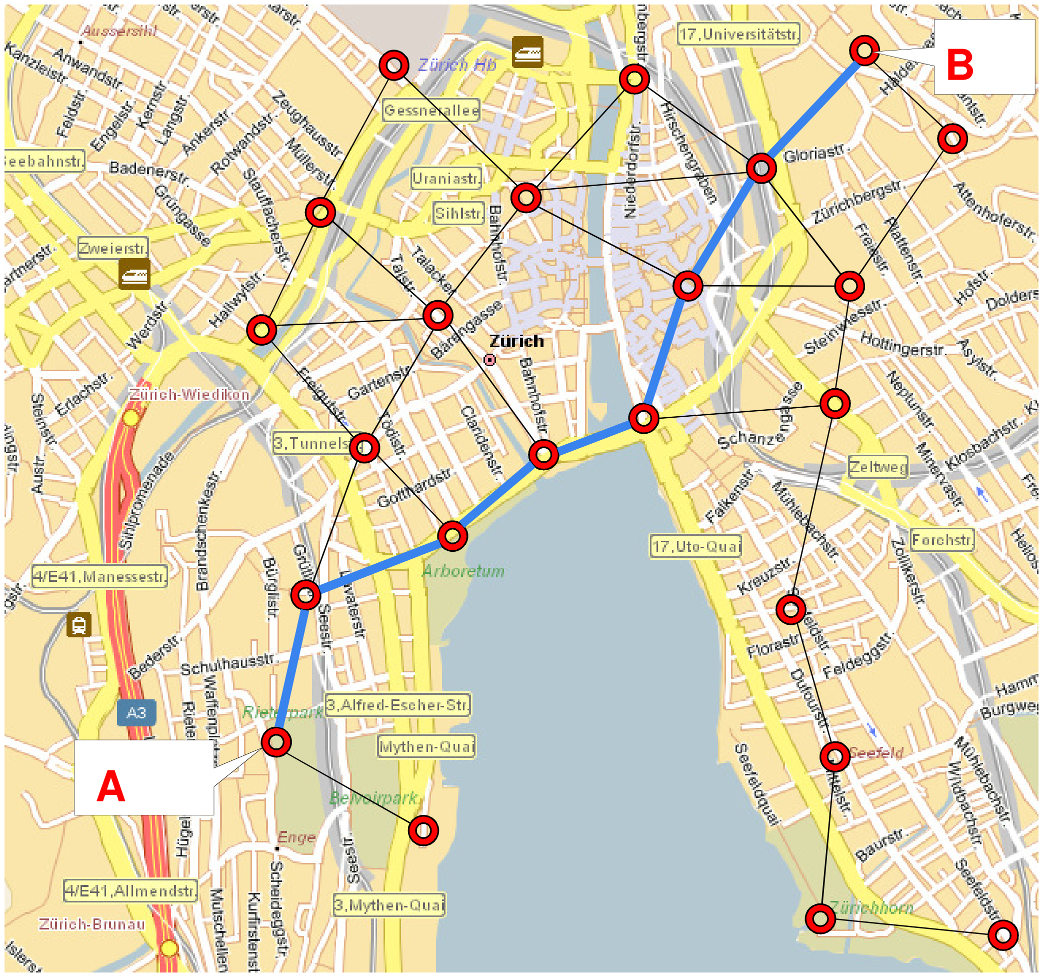
*backtracking? Does this mean that we need a stack?!?



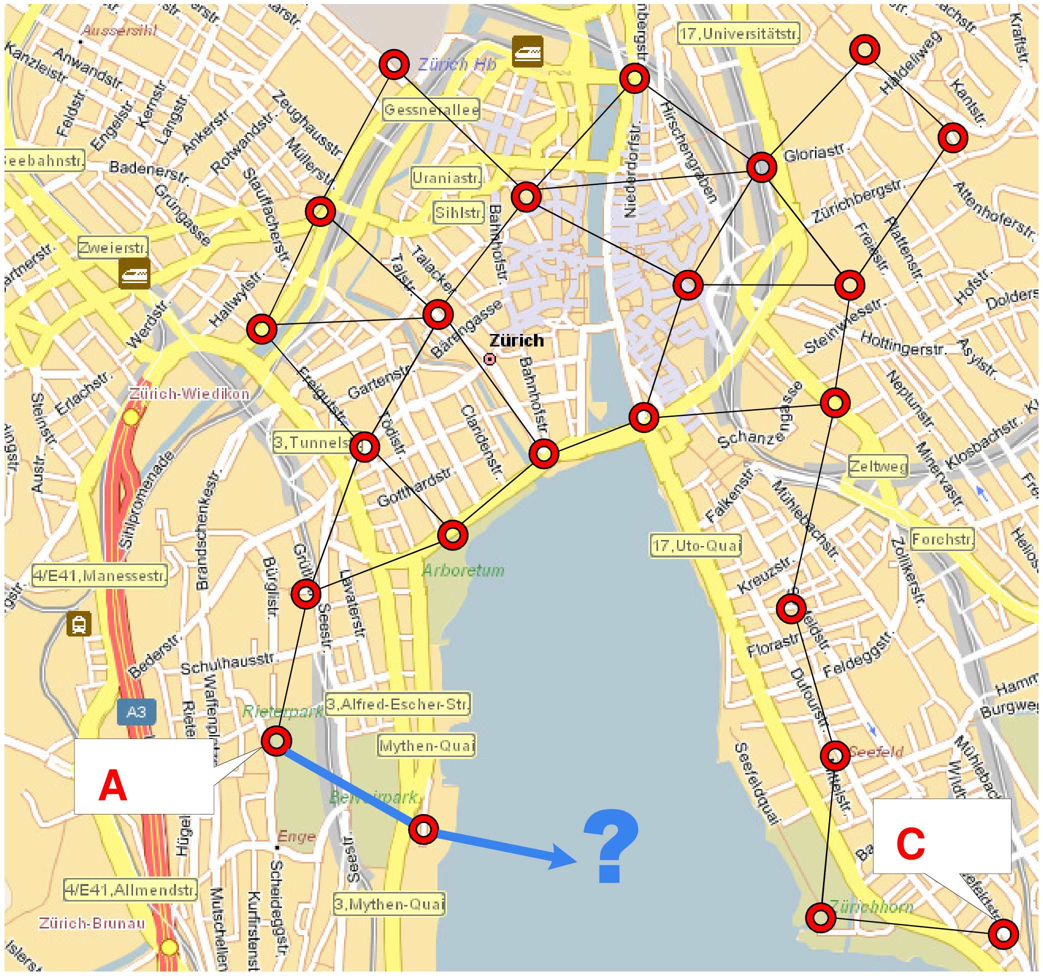
Geo-Routing: Strictly Local



Greedy Geo-Routing?



Greedy Geo-Routing?

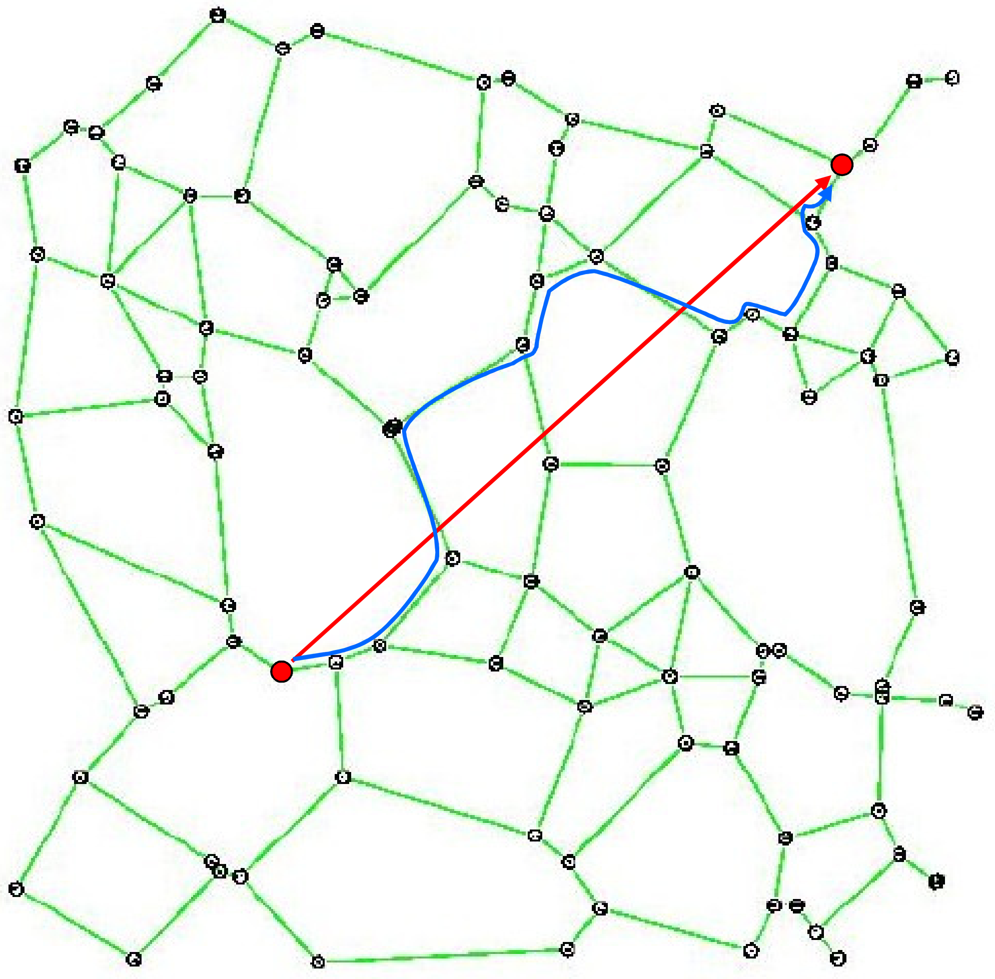


What is Geographic Routing?

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

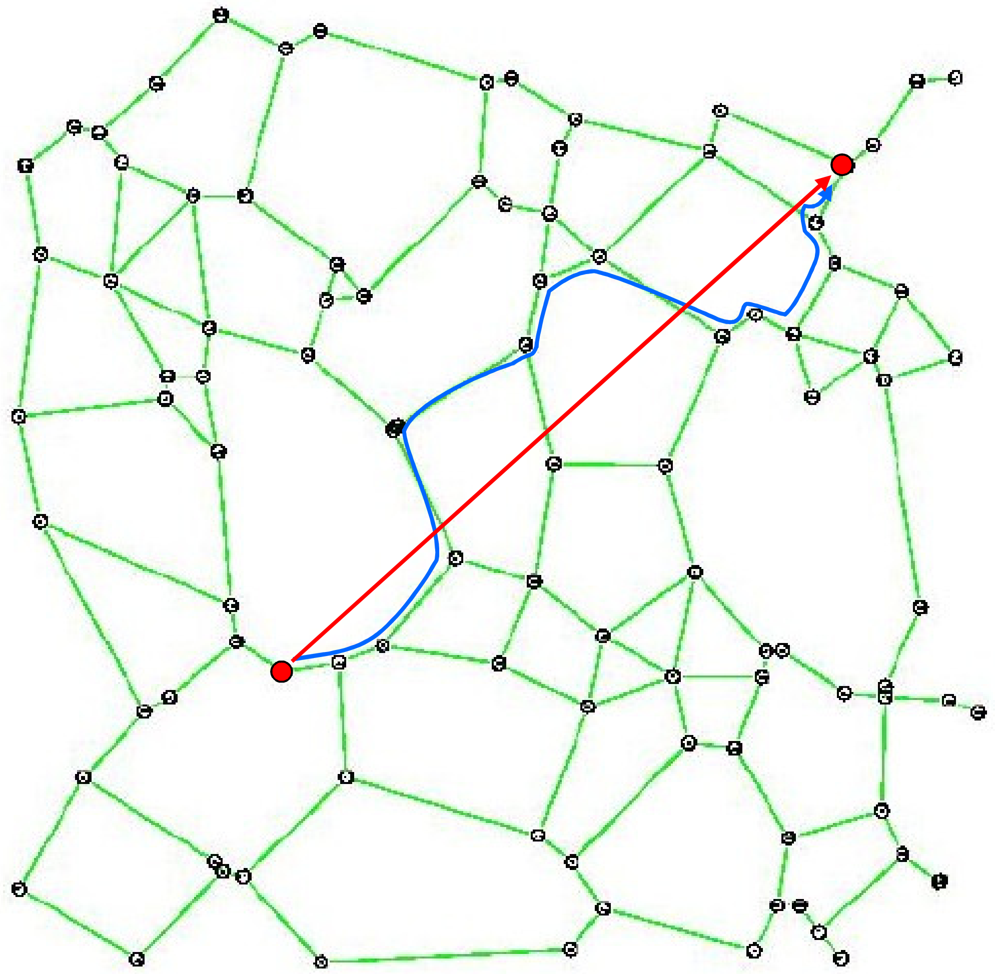
Greedy routing

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



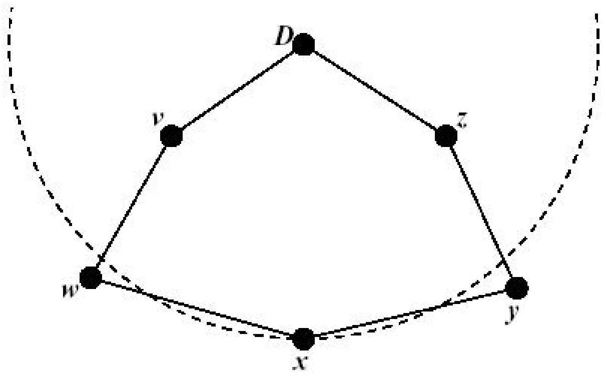
Greedy routing

- 0. Start at *s*.
- 1. Proceed to the neighbor closest to *t*.
- 2. Repeat step 1 until either reaching *t or a local minimum with respect to* the distance from *t, that is a node v without any neighbor closer to t than v itself.*

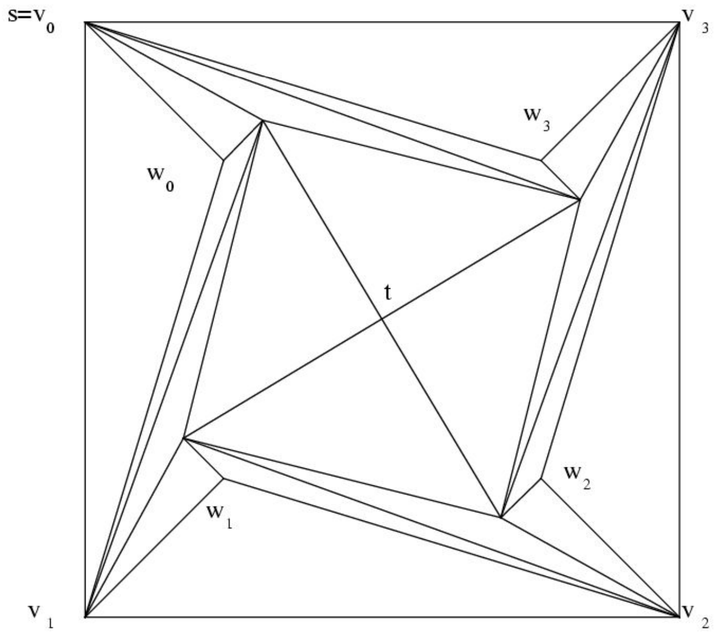


Examples why greedy algorithms fail

- 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 $v_0, w_0, v_1, w_1, \ldots, v_3, w_3, v_0, \ldots$

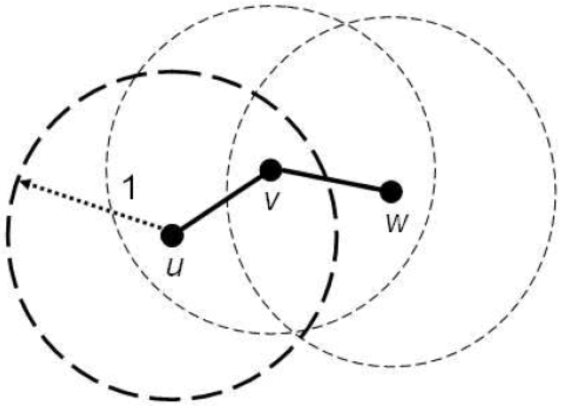


Euclidean and Planar Graphs

- Euclidean: Points in the plane, with coordinates, e.g. UDG

- UDG: Classic computational geometry model, special case of disk graphs.

- All nodes are points in the plane, two nodes are connected iff (if and only if) their distance is at most 1, that is $\{u,v\} \in E$, $|u,v| \leq 1$

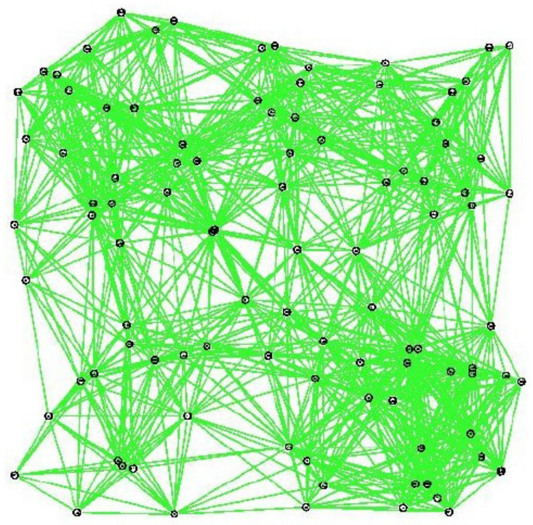


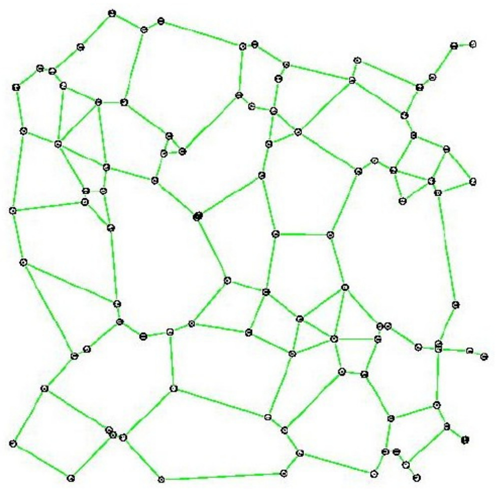
+ Very simple, allows for strong analysis

– Particularly bad in obstructed environments (walls, hills, etc.)

Euclidean and Planar Graphs

- Planar: can be drawn without "edge crossings" in a plane

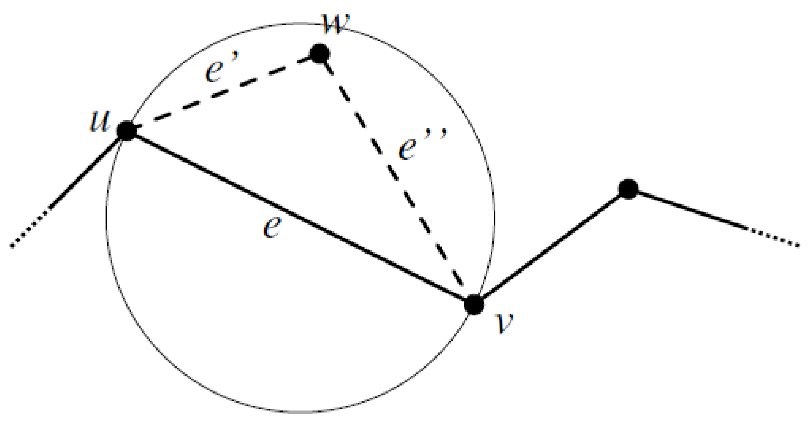




- A planar graph already drawn in the plane without edge intersections is called a plane graph.
- Now we will see how to make a Euclidean graph planar.

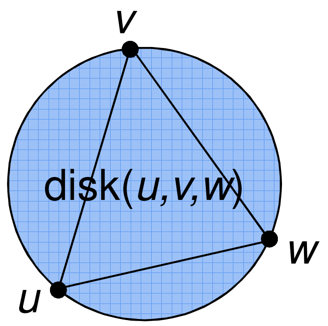
Euclidean and Planar Graphs

- In order to achieve planarity on the unit disk graph G, the Gabriel graph is employed.
- A Gabriel graph contains an edge between two nodes u and v iff the disk (including boundary) having uv as a diameter does not contain a "witness" node w.

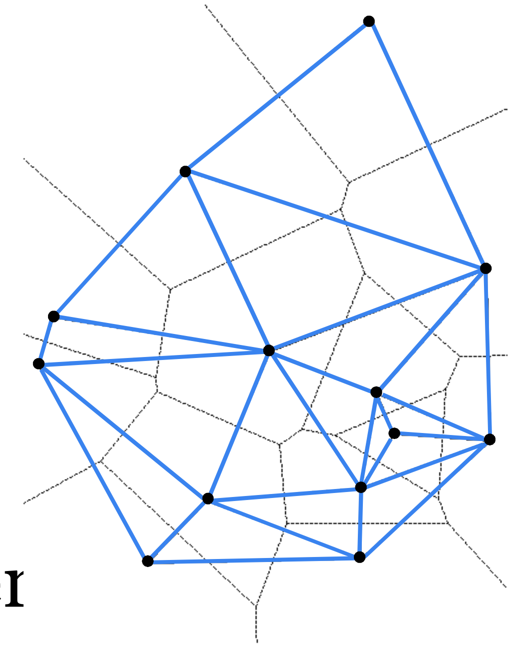


Delaunay Triangulation

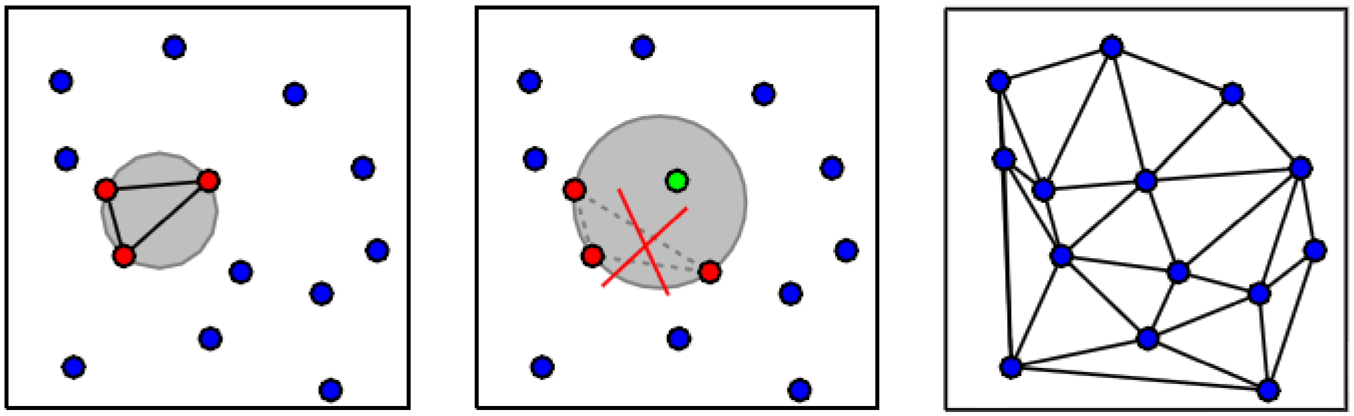
- Let disk(u,v,w) be a disk defined by the three points u,v,w.
- The Delaunay Triangulation (Graph) DT(V) is defined as an undirected graph (with E being a set of undirected edges). There is a triangle of edges between three nodes u,v,w iff the disk(u,v,w) contains no other points.



- The Delaunay Triangulation is the dual of the Voronoi diagram, and widely used in various CS areas
  - the DT is planar
  - the DT is a geometric spanner

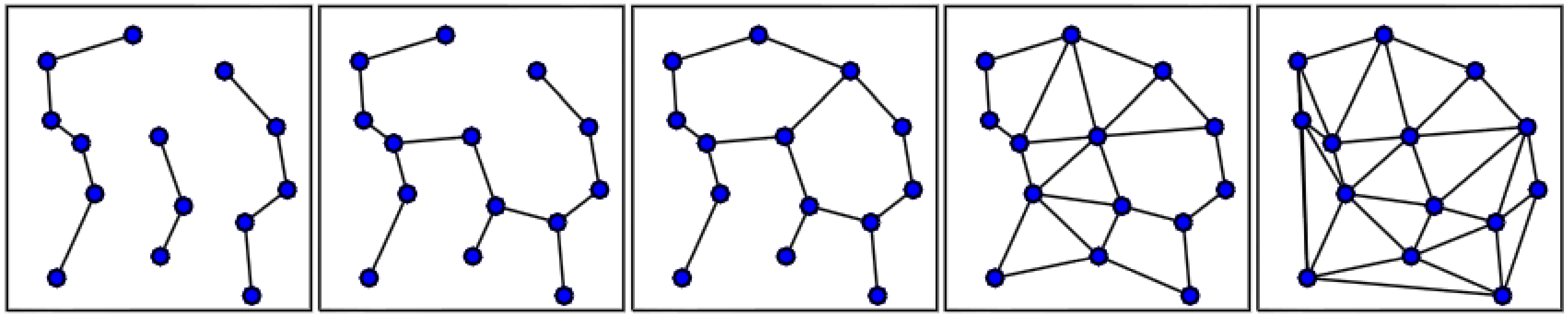


Delaunay Triangulation


(a) triangle (b) not a triangle (c) resulting graph

Properties of Proximity Graphs

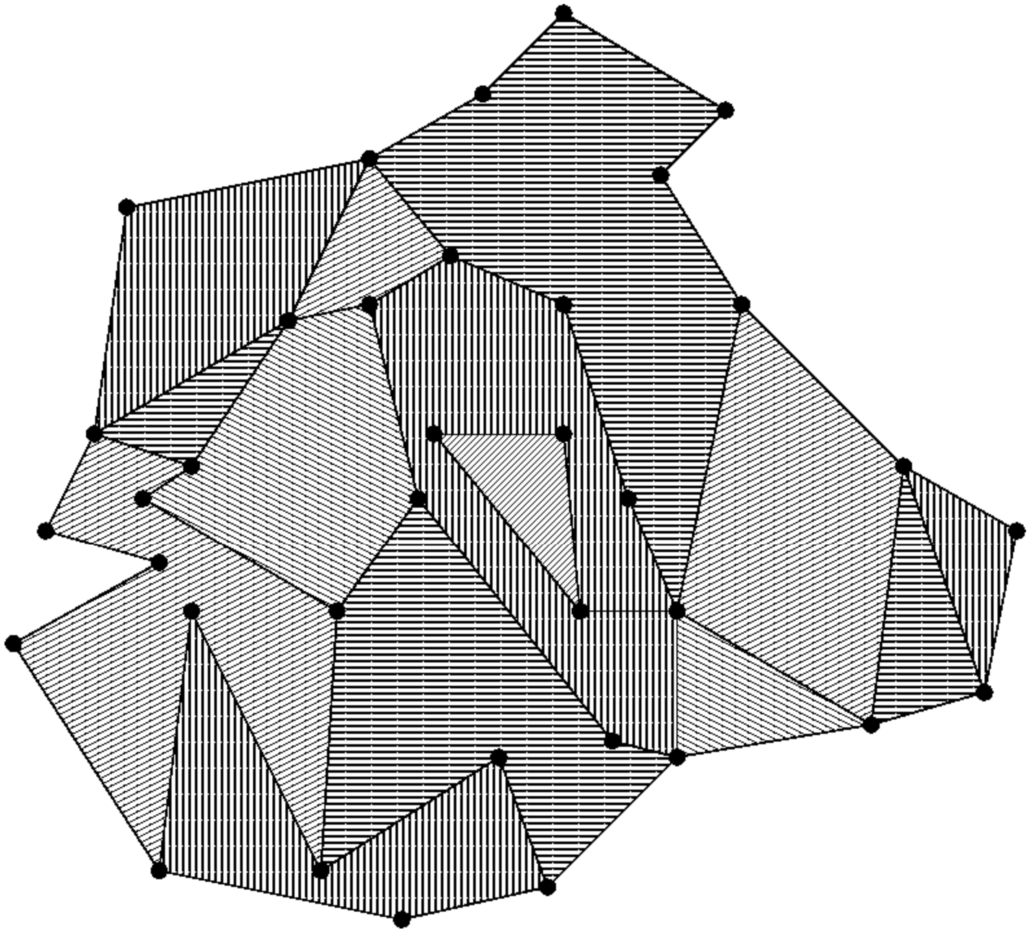
- Theorem 1:
  MST $\subseteq$ RNG $\subseteq$ GG $\subseteq$ DT


(a) NNG (b) EMST (c) RNG (d) GG (e) DT

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

Breakthrough idea: route on faces

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

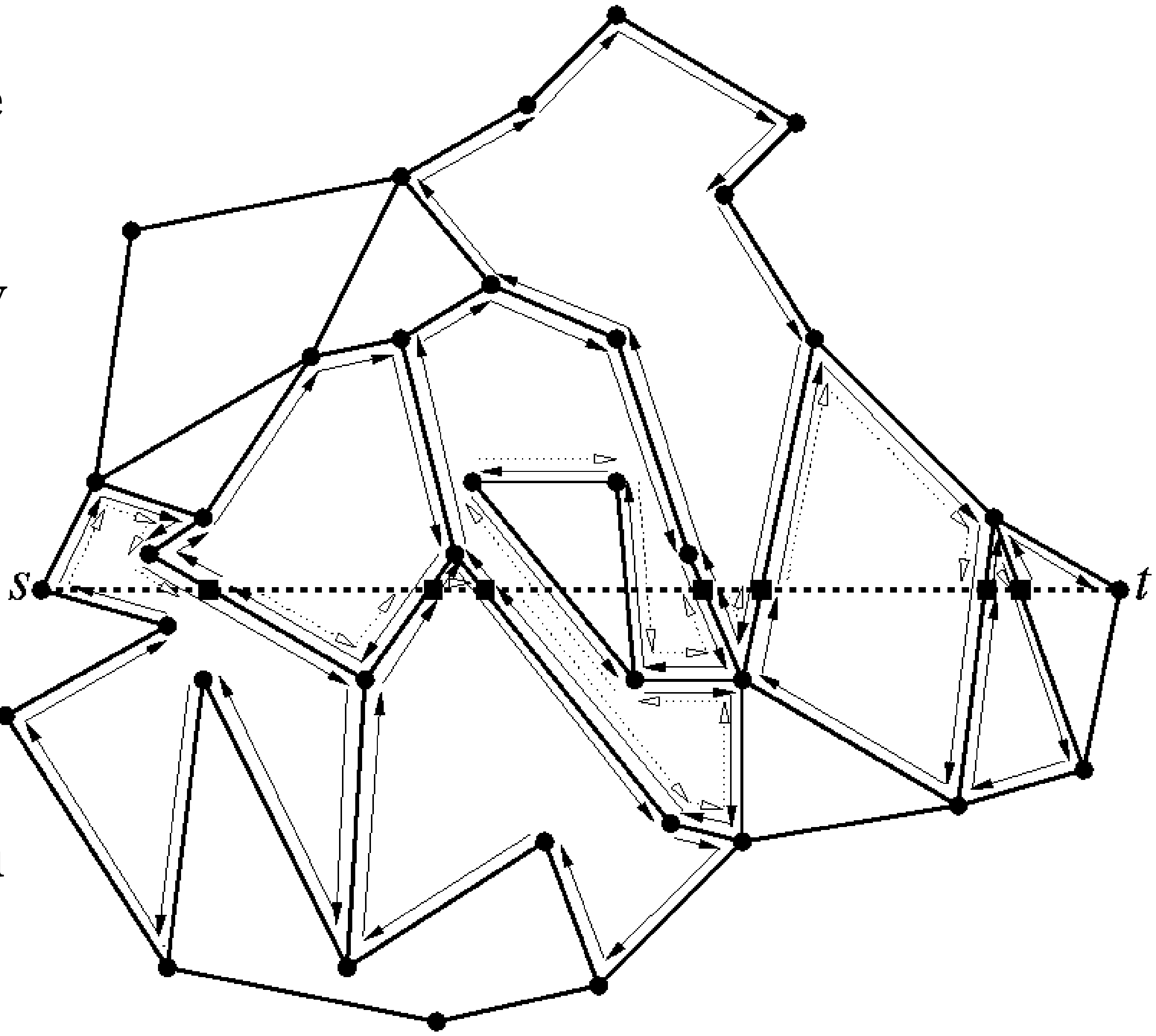


- Kranakis, E., Singh, H., Urrutia, J.: **Compass routing on geometric networks,** in proc. of the 11th CCCG, Vancouver, Canada, pp. 51–54 (1999)

Face Routing

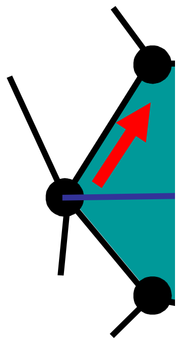
0. Let f be the face incident to the source s, intersected by (s,t)

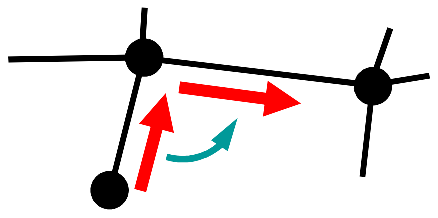
1. Explore the boundary of f; remember the point p where the boundary intersects with (s,t) which is nearest to t; after traversing the whole boundary, go back to p, switch the face, and repeat 1 until you hit destination t.



Face Routing Properties

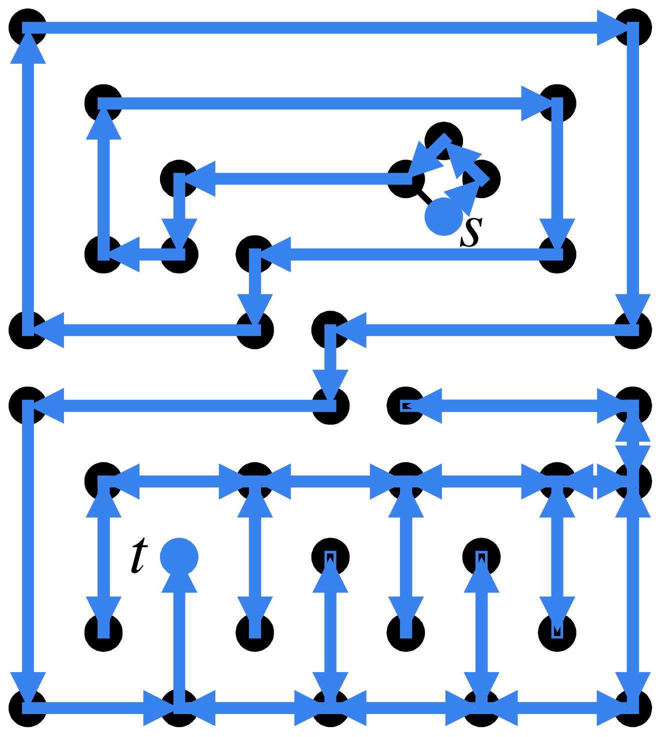
- All necessary information is stored in the message
  - Source and destination positions
  - Point of transition to next face
- Completely local:
  - Knowledge about direct neighbors‘ positions sufficient
  - Faces are implicit





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

Face Routing Works on *Any* Graph



Face routing is correct

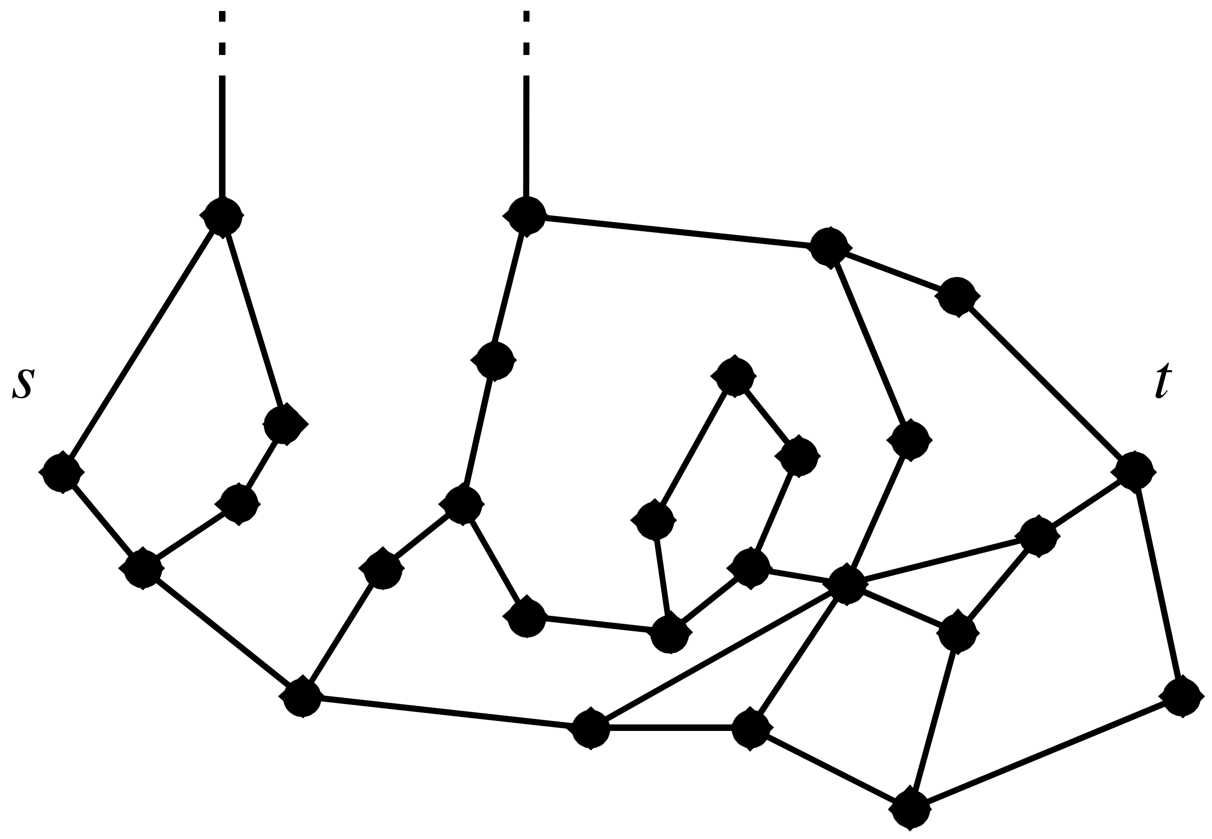
- **Theorem:** Face routing terminates on any simple planar graph in $O(n)$ steps, where n is the number of nodes in the network

- 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.

- **Euler's formula gives** $v - e + f = 2$.
- From $v - e + f = 2$ and $2e >= 3f$ (one face has minimum 3 edges and each edge has maximum two faces)
- $e \leq 3v - 6$ if $v \geq 3$.

Face Routing

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



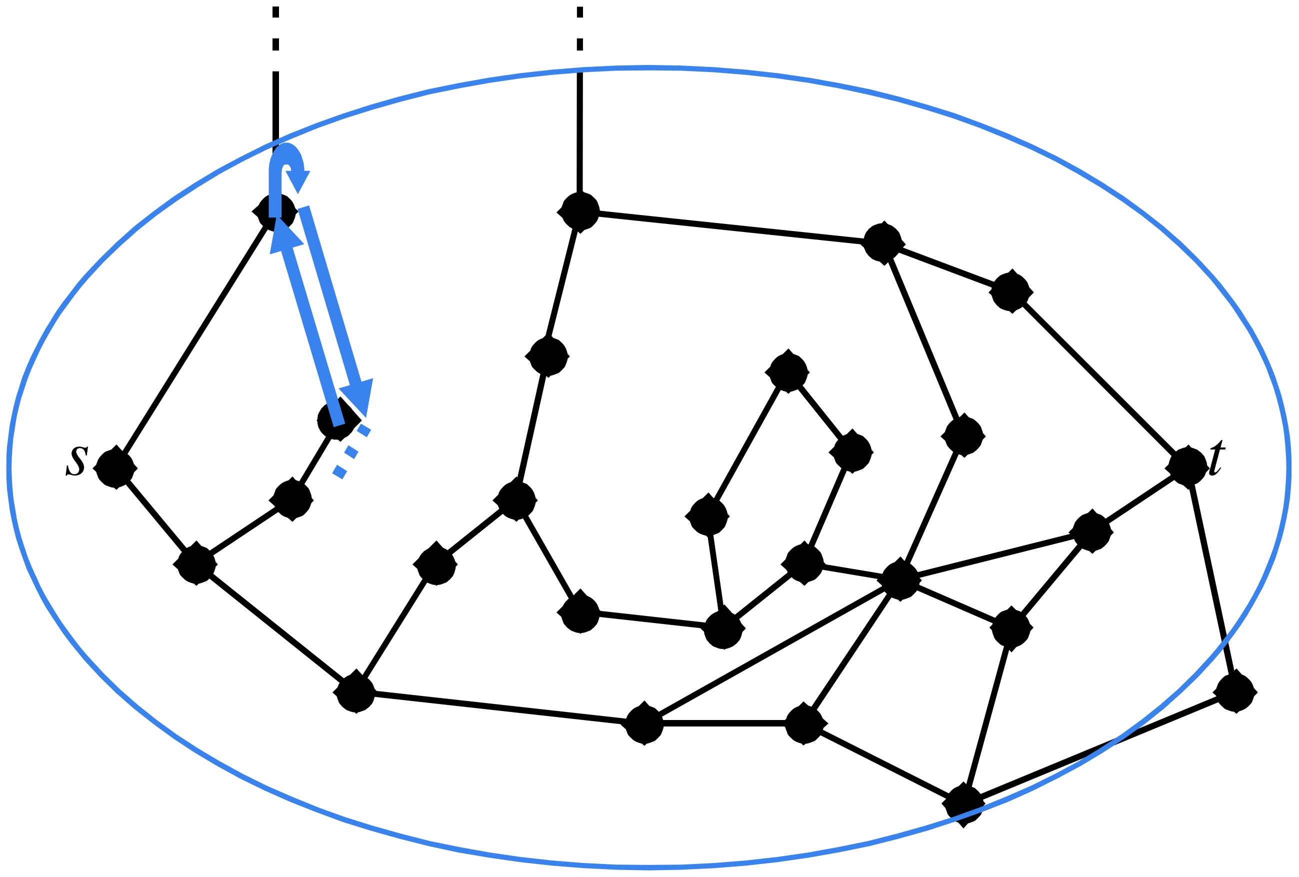
Is there something better than Face Routing?

How can we improve Face Routing?

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).

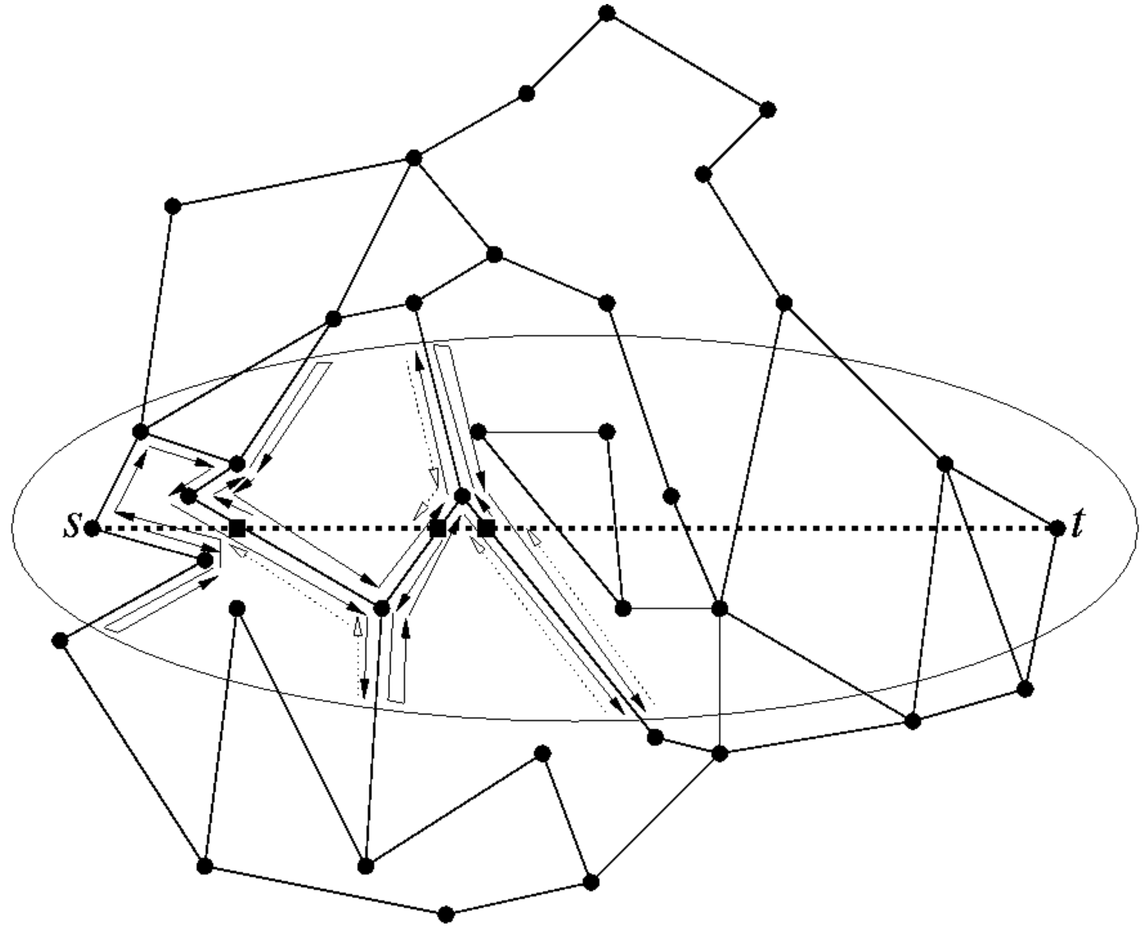
Bounding Searchable Area



Adaptive Face Routing (AFR)

- Idea: Use face routing together with "growing radius" trick:

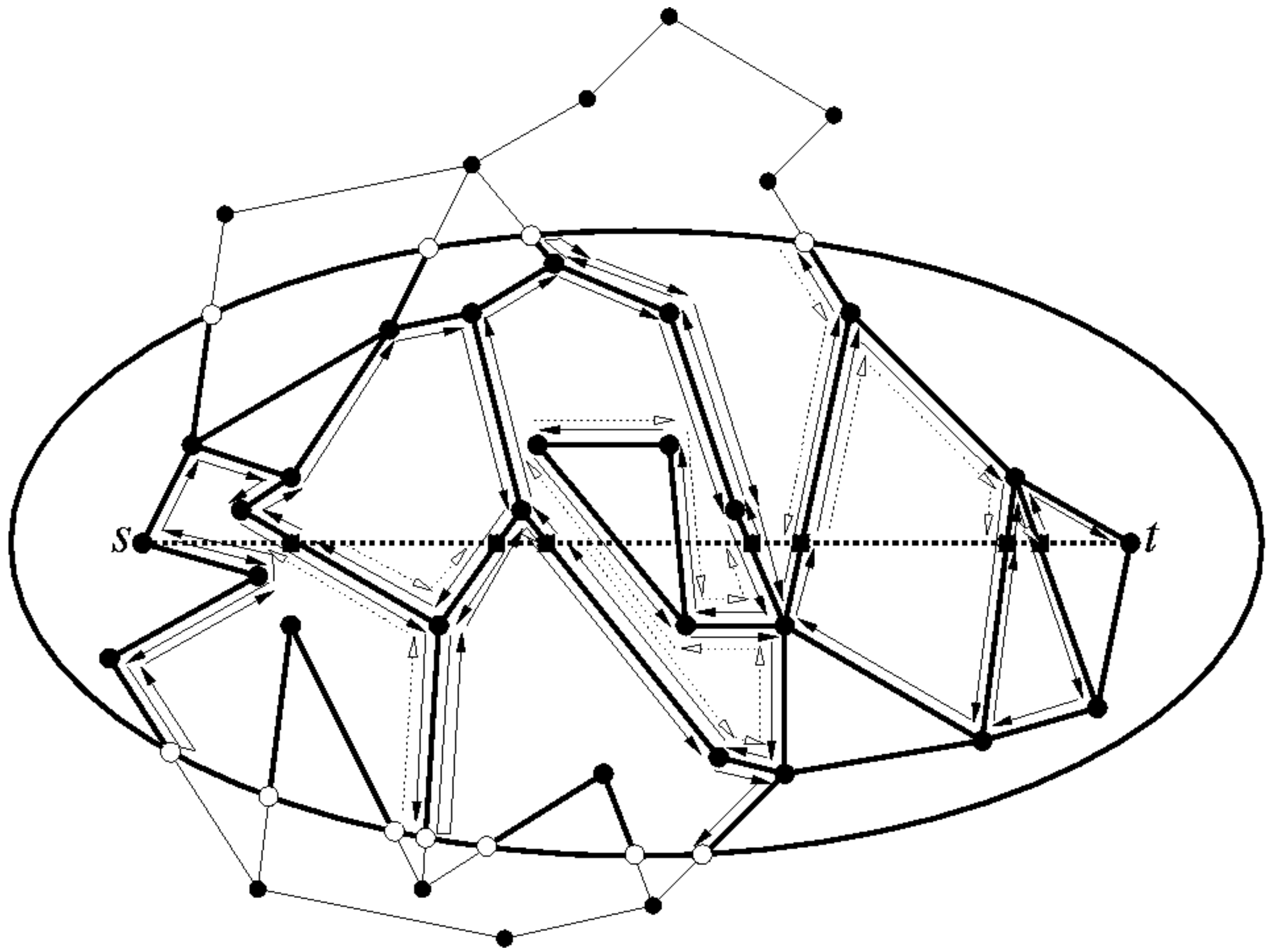
- That is, don't route beyond some radius r by branching the planar graph within an ellipse of exponentially growing size.



- Kuhn, F., Wattenhofer, R., Zollinger, A.: **Asymptotically optimal geometric mobile ad-hoc routing**, 6th Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Atlanta,USA (2002)

AFR Example Continued

- We grow the ellipse and find a path



AFR Pseudo-Code

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

1. Nodes w 2 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 reducing 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.)

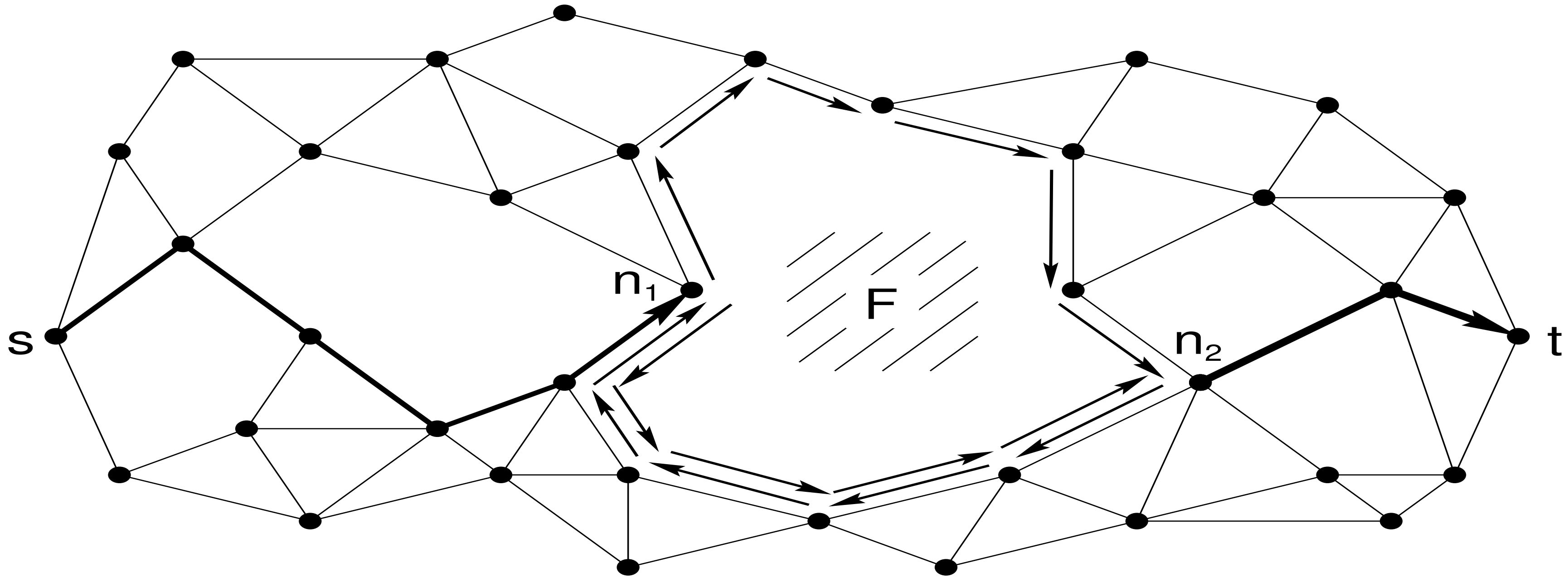
2. If step 1 did not succeed, double c and go back to step 1.

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

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 point closest to destination

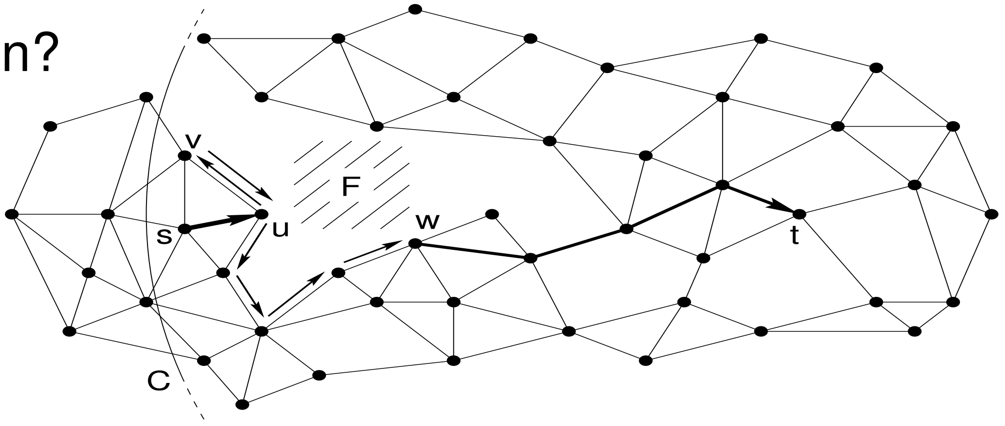


GOAFR+ – Greedy Other Adaptive Face Routing

- 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 > σ ¢ q (constant σ > 0)

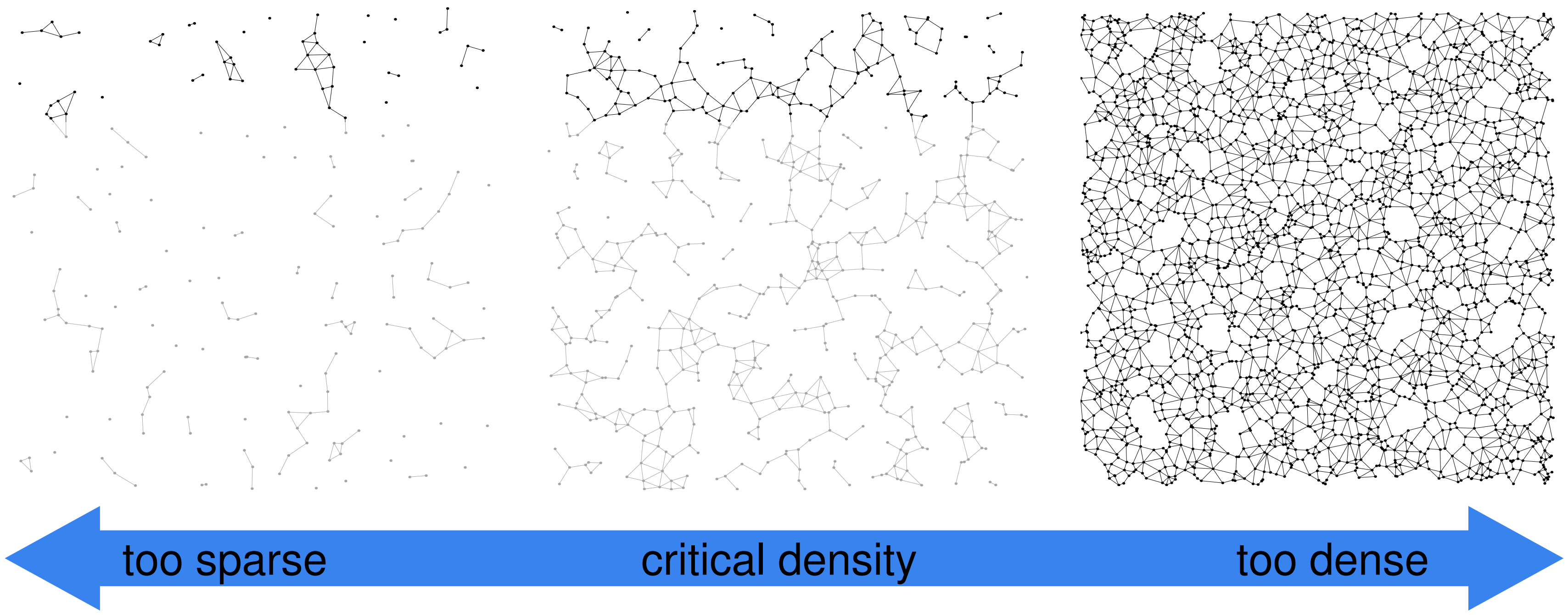
Theorem: GOAFR is still asymptotically worst-case optimal…
…*and* it is efficient in practice, in the average-case.

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



Average Case

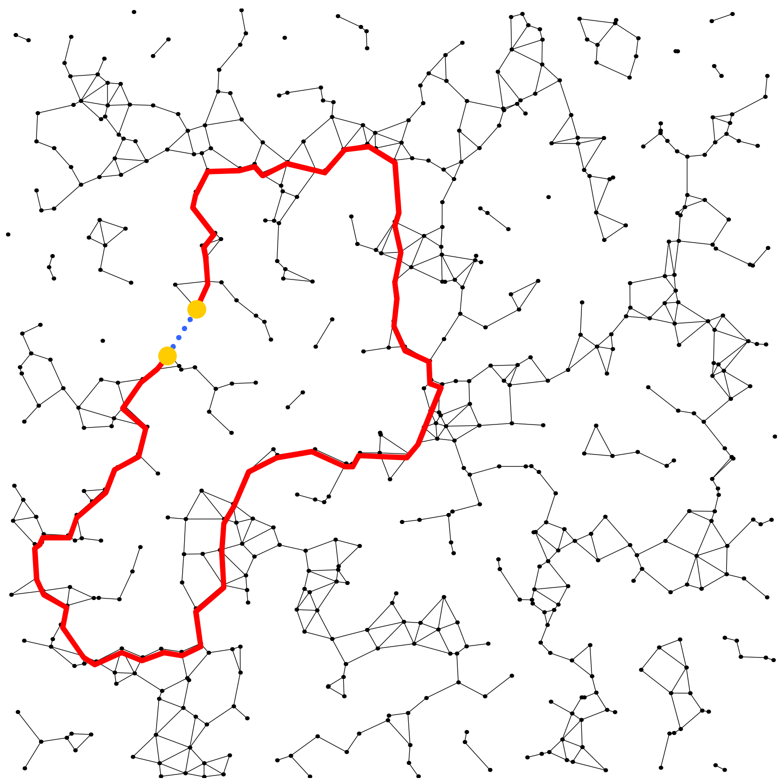
- Not interesting when graph not dense enough
- Not interesting when graph is too dense
- Critical density range ("percolation")
  - Shortest path is significantly longer than Euclidean distance



Critical Density: Shortest Path vs. Euclidean Distance

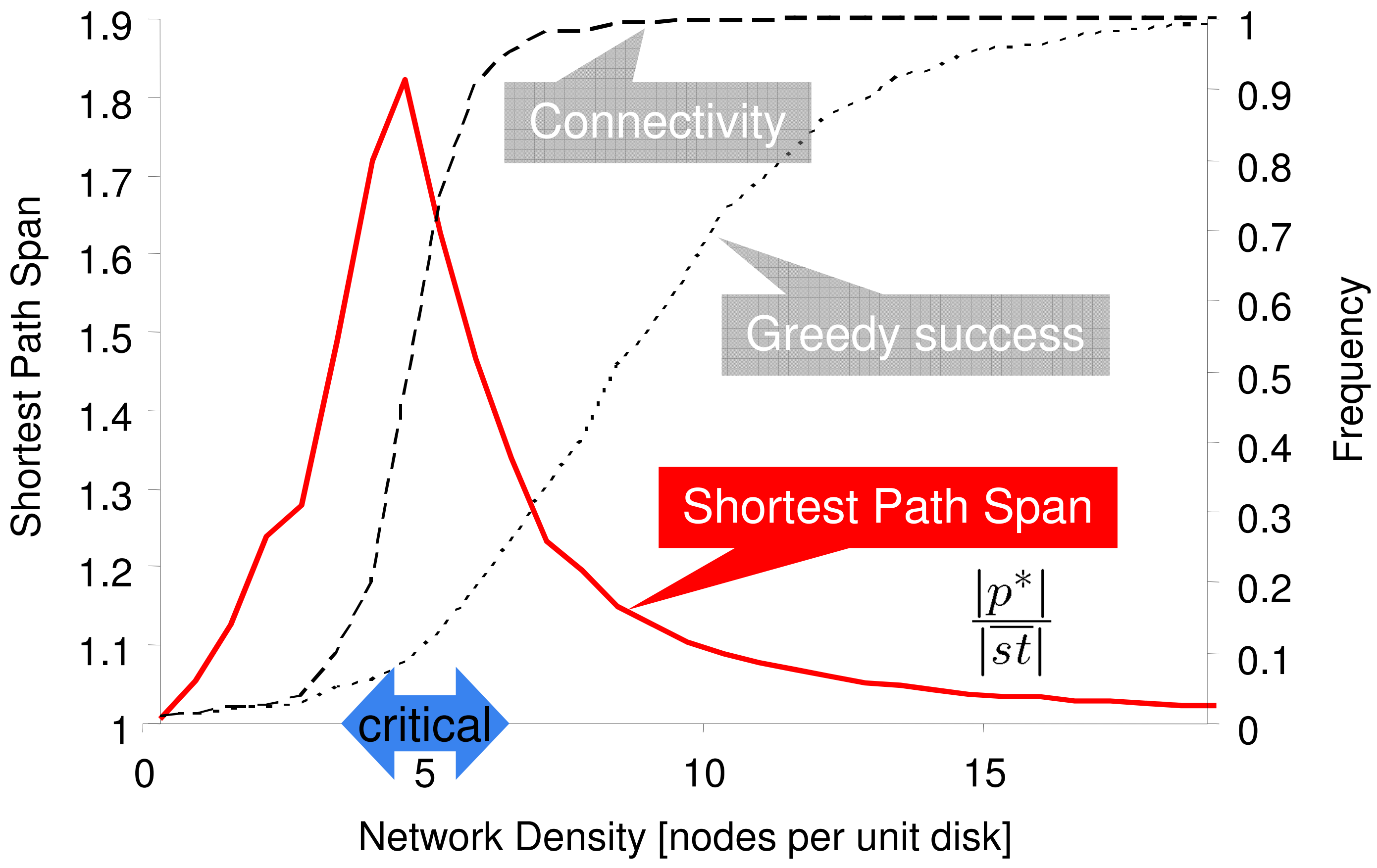
- Shortest path is significantly longer than Euclidean distance

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

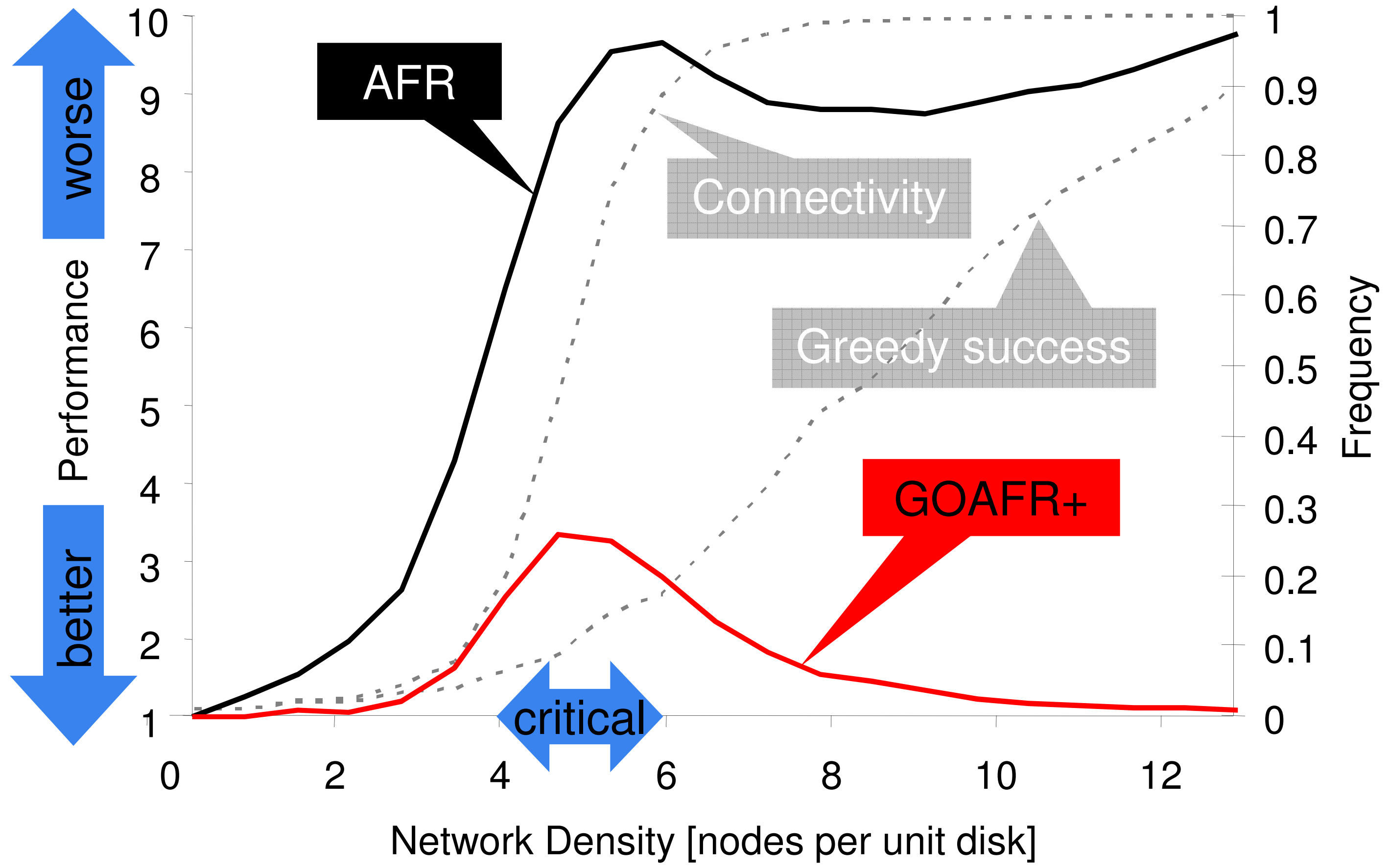


- Critical density range mandatory for the simulation of any routing algorithm (not only geographic)

Randomly Generated Graphs: Critical Density Range



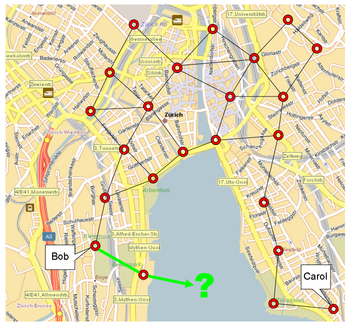
Simulation on Randomly Generated Graphs



A Word on Performance

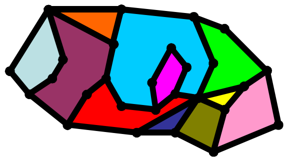
- What does a performance of 3.3 in the critical density range mean?
- If an optimal path (found by Dijkstra) has cost c, then GOAFR+ finds the destination in 3.3¢c steps.
- It does *not* mean that the *path* found is 3.3 times as long as the optimal path! The path found can be much smaller…
- Remarks 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

GOAFR: Summary

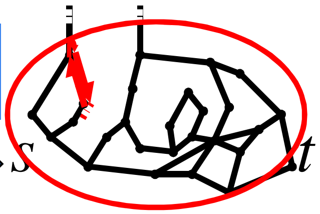


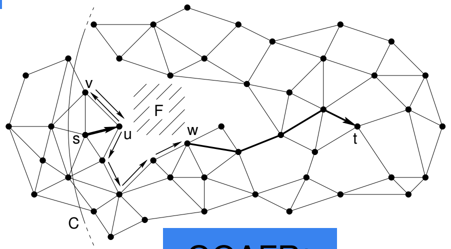
Greedy Routing

Face Routing



Adaptive Face Routing





GOAFR+

Average-case efficiency

“Practice”

Worst-case optimality

“Theory”

Routing with and without position information

- Without position information:
  - Flooding
  - Distance Vector Routing

- 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

Summary

- If position information is available geo-routing is a feasible option.
- Face routing guarantees to deliver the message.
- Combining greedy and face gives efficient algorithm.
- Even if there is no position information, some ideas might be helpful.

- Geo-routing is probably the only class of routing that is well understood.
- There are many adjacent areas: topology control, location services, routing in general, etc.

Open problem

- Geo-routing is one of the best understood topics. In that sense it is hard to come up with a decent open problem.
- Open problem: How much information does one need to store in the network to guarantee only constant overhead?
  - Variant: Instead of UDG some more realistic model
  - How can one maintain this information if the network is dynamic ?