Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2010

Prof. Dr. Roger Wattenhofer, Philipp Sommer, Johannes Schneider

Ad Hoc And Sensor Networks
Sample Solution to Exercise 2

Assigned: October 4, 2010
Due: October 11, 2010

1 Face Routing in Theory

a) In the graph of Figure 1[1] planarization is essential for the face routing algorithm to succeed. The graph is constructed such that $|ct| \leq 1$, $|ab| \leq 1$, and $|at| = |bt| > 1$.

If we consider a pure face routing, and that the message is sent from s' to t, the algorithm first visits the face indicated by the dashed arrow. It finds the edge (a, b) to be the closest one which intersects $\overline{s't}$, and visits again the same face (infinitly often) since the other face is further away from t.

Also a "stupid" face routing algorithm which does not check if the new face is closer to t cannot find the destination in this case. Once again the algorithm first visits the face indicated by the dashed arrow. Edge (a, b) is found to be the closest intersection with $\overline{s't}$, and the algorithm switches to the face defined by a, b, and d. It then once again selects edge (a, b) as the closest intersection with $\overline{s't}$ and returns to the initial face.

In a GFG (greedy - face - greedy) routing algorithm, face routing starts at node s' when the message is sent from s to t. The face traversal visits the nodes as indicated by the dashed line, skipping node c, and therefore not finding t. When s' is the node closest to t (except node c), face routing fails to find a node closer to t and drops the message.

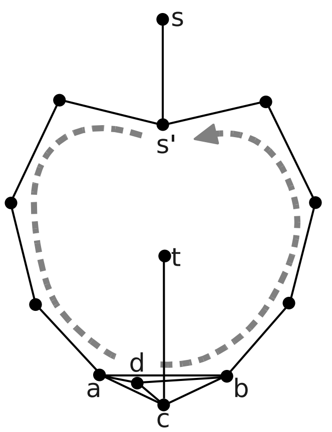

Figure 1: Planarization is essential for face routing.

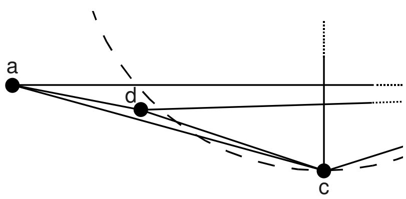

Figure 2: Close-up of the placement of node d. The dashed line is the unit circle around t

b) If d is too small, nodes cannot learn locally about neighboring nodes: Node A has no means to determine the existence of node C and D in Figure 3. In the graph shown on the right

[1]Please note that the graph is not true to scale as nodes a and d are very close together. Figure 2 shows a close-up of the node placement necessary to fulfill the UDG properties.

of Figure 3, B needs 13 hops to learn about the existence of C or D – and we can grow this graph arbitrarily to increase the required number of hops to detect the crossing. Therefore, no local algorithm can planarize a QUDG with $d < 1/\sqrt{2}$.

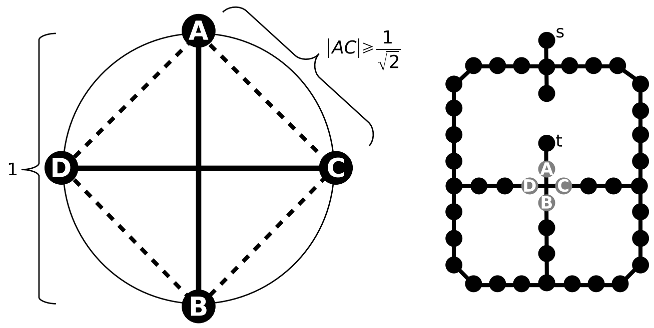

Figure 3: QUDG with $d < 1/\sqrt{2}$ cannot be planarized with local information.

c) We can observe that intersecting edges can be detected locally if $d \geq \frac{1}{\sqrt{2}}$. We use the following fact:

Fact 1: For $d \geq \frac{1}{\sqrt{2}}$ we have for any edge $e_1 = (u_1, v_1)$ that is intersected by an edge $e_2 = (u_2, v_2)$ that either u_1 or v_1 is connected to u_2 or v_2.

Proof: We have to show that at least one of the four sides of the quadrangle (u_1, u_2, v_1, v_2) is shorter than $\frac{1}{\sqrt{2}}$. Because the sum of the interior angles in the quadrangle is 2π, at least one of the angles is at least $\pi/2$. W.l.o.g. assume that the interior angle at u_2 is greater or equal to $\pi/2$. In this case, u_2 lies in the disk with diameter (u_1, v_1), Therefore, $|u_1 u_2| \leq \frac{1}{\sqrt{2}}$ or $|v_1 u_2| \leq \frac{1}{\sqrt{2}}$, which shows the claim.

Thus, it suffices if each node sends its neighbor-list to all of its neighbors such that all edge-intersections can be detected.

To planarize the graph, we now introduce a *virtual node* at each intersection point, which subdivides the original edges. These virtual edges can be handled by the adjacing physical edges. Note that the obtained graph is not anymore a UDG or QUDG, but it is planar, which suffices for face routing.

2 Face Routing in Practice

a) For the pure face routing algorithm, we need to store the line $\overline{st}$ as well as p, the closest intersection point of the current face with $\overline{st}$. To deliver the message, we may also store the ID of t.

For a GFG algorithm, we need to store at least the following state information:

- Position (and ID) of the destination
- Flag indicating whether the message is in face routing mode or in greedy forwarding mode
- While in face routing mode, the edge which the message visited first to detect the second round in the same face. *Note*: It is not sufficient to store the node on which we started the face traversal. (Why?)
- While in face routing mode, a flag indicating whether the message is in its first or second round in the current face
- While in face routing mode, the position of the node closest to the destination, which we have seen so far on the current face

b) If the next face to visit equals the current face, the message got stuck in an infinite loop and won't be able to reach the destination. This happens only when the destination is disconnected.

c) There are many solutions to this question. Most importantly, a realistic routing algorithm should not rely too much on the planarity and stability of the underlying network graph. E.g. it may perform a planarization of the graph and perform greedy-face-greedy (GFG) routing. In addition, the algorithm should detect/guess cycles and other network changes and switch to a more primitive routing version in such a situation, e.g. flooding or random walks.