Algorithmic aspects of communication networks	**Lecture 9**
OR 739 Spring 1997	**February 10, 1997**

1 Routing on the $n \times n$ mesh

In this lecture we consider a special case of the on-line admission control problem. Our goal is to achieve the same $O(\log m)$ competitive ratio while removing the assumption that each edge is permitted be used $k = \Omega(\log n)$. Kleinberg and Tardos give a version of admission control that works in *densely embedded graphs* for any k. In this lecture we consider the special case where the graph is the $n \times n$ mesh and $k = 5$.

First we partition the $n \times n$ mesh into smaller sub-meshes of size $r \times r$, where $r = \Theta(\log_2(2m))$ and $m = 2n(n+1)$ is the number of edges in the original mesh. (We may also need some meshes of size between $r \times r$ and $2r \times 2r$ if our original mesh does not break up evenly.) For simplicity, in this lecture we assume that for each connection request, the terminals s and t are in different sub-meshes. (Alternatively we make the stronger assumption that each pair of terminals is at least $4r$ apart.)

We construct the *aggregated network* by contracting all nodes within a sub-mesh into a singe supernode (replacing parallel edges by a single edge whose capacity is the number of parallel edges that it replaced). The aggregated network is a rough approximation of our original network. Observe that since $k = 5$ every edge in the aggregated network has capacity at least $5r$. Thus we will be able to apply our previous admission control algorithm, since now our edge capacity assumption is satisfied.

We will consider admission control and routing in two steps. The first step will involve the aggregated network. The second will use the original network. Of course, ultimately, we need to route in the original network, but the routing in the aggregated network will guide us.

Admission decision for pair (s, t)

1. Greedily route from s to the boundary of its sub-mesh
2. Greedily route from t to the boundary of its sub-mesh
3. Run admission control in aggregated network

In each of the 3 cases, we restrict ourselves to use each original edge at most once. (A feasible solution may use each edge up to 5 times, but we save 4 of these for later use.) This means that each edge of the original network may be used at most once for the greedy routes of steps 1 and 2; and for the admission control in the aggregated network we use each edge at most a factor of 5 times less than its capacity. By greedy in steps 1 and 2, we mean find any path from the terminal to the boundary using only unused edges. If all 3 attempts succeed then we will accept the requested pair (s, t). Otherwise we reject.

Routing decision for pair (s, t)

1. Route s and t to their sub-mesh boundaries as above

2. Connect sub-mesh boundary points from aggregated network routing in original network

3. Connect the first and last of the aggregated network boundary points to the respective boundary points in step 1, if necessary

Note that in the second two steps, the terminals pairs are on the boundary of a sub-mesh. The following lemma says that routing these connections is easy.

Lemma 1 *On the $r \times r$ mesh, any set of connections from the boundary of the mesh, such that every boundary terminal is used at most once, can be done using every edge at most 3 times.*

Proof. We can do this greedily using only *1-bend* and *2-bend* routes. If s and t are on adjacent sides of the boundary then we connect them using 1-bend. (For example, if s in on the left boundary and t is on the top boundary, then we route from s horizontally along s's row to the column for t, then vertically up t's column to t.) Each of the edges in the row (column) for s and the column (row) for t may be used at most once.

If s and t are on the same side or opposite sides then we use a 2-bend route. (For example, if s is on the left side and t is on the right side, then we route from s horizontally along s's row to a *linking* column, then vertically up or down the column to t's row, then horizontally along t's row to t.) Each of the edges in the row (column) for s and the row (column) for t may be used at most once. Additionally, we might also need to use edges in another *linking* column (row).

Thus, if we disregard the part of the 2-bend routes that use the linking columns and rows, each edge is used at most twice, once by each of the connections whose terminal is in the two ends of the row or column of the edge. If for every 2-bend request we choose our linking columns and rows to be distinct, then our routing uses each edge at most 3 times. ■

Note that we can do this routing in an on-line setting. Also, the same argument works even if the mesh has size between $r \times r$ and $2r \times 2r$. But now we might not be able to choose the linking columns and rows to be distinct. (For example, if we have an $2r \times r$ mesh, and there are $2r$ requests from the left boundary to the right boundary then we might need $2r$ linking columns for the 2-bend routes. But there are only r columns available.) However, we can choose the routing so that the linking columns and rows are never used more than twice in that role. Thus, we might use an edge as many as 4 times.

Lemma 2 *The algorithm produces a feasible routing for the accepted pairs.*

Proof. Each edge is used at most 1 time in routing to the boundary. Thus, a sub-mesh uses each of its boundary nodes at most once and the previous lemma implies that each edge is used at most 4 times in routing between boundary nodes. By assumption, we are permitted to use each edge 5 times. ■

Let $\mathcal{A}$ denote set of pairs accepted by our algorithm. Let $\mathcal{O}$ denote of pairs accepted in some optimal off-line solution. Let $\mathcal{O} - \mathcal{A} = \mathcal{O}_g \cup \mathcal{O}_l$ where $\mathcal{O}_g$ is the set of pairs rejected by global routing on aggregated network but accepted by optimal and $\mathcal{O}_l$ is the set of pairs rejected by local routing to boundary but accepted by optimal.

Theorem 1 $|\mathcal{O}_g| \leq 5\log_2(2m)|\mathcal{A}|$

Proof. The proof is almost identical to the admission control proof from the first lecture. However, our algorithm assumes the capacity of every original edge is $k = 1$ while the optimal solution may use each original edge $k = 5$ times. We lose a factor of 5 in the analysis since the optimal solution may use every edge a factor of 5 times more than our algorithm. ■