Algorithmic aspects of communication networks	**Lecture 4**
OR 739 Spring 1997	**January 31, 1997**

In this lecture we consider an algorithm for the Load Balancing problem that was introduced in the last class. Recall the problem description:

Load Balancing Problem

- We are given an undirected graph $G = (V,E)$ with n nodes and m edges.
- Pairs of nodes (s,t) arrive online.
- When a pair (s,t) arrives, we are **required** to connect the pair via some s-t path in G. (In contrast to admission control, we do **not** have the option of rejecting the pair.)
- The objective is to minimize (over all edges) the number of paths that use an edge. That is, we wish to min $(\max_{e\in E} f(e))$, where $f(e) \equiv$ the number of paths using edge e. Typically, $\max_{e\in E} f(e)$ is called the *congestion* of the network.

The algorithm we will consider (hereafter referred to as Algorithm A) assigns to each edge $e\in E$ a cost $c(e)$. These edge costs are used to determine the path along which an arriving (s,t) pair will be routed and are updated after each pair is routed. Specifically, Algorithm A works as follows:

Algorithm *A*

1. $c(e) \equiv (3/2)^{f(e)} \quad \forall\ e\in E$
2. When a pair (s,t) arrives, connect s and t via a minimum cost s-t path in G, where the path cost is computed with respect to the cost function c.

Note that the edge costs $c(e)$ are implicitly updated each time a pair (s,t) is routed since $f(e)$ increases by 1 for each edge e in the s-t path that was used.

Since Algorithm A (as we have defined it) has no "stopping time", any statements made about its performance must be with respect to a particular point in time. The statements that we will make regarding the congestion achieved by Algorithm A (versus the "optimal" congestion) will hold for **all** points in time after the algorithm begins. However, to reduce ambiguity in our analysis, we will pick a particular point in time, time T, and compare the congestion of Algorithm A at time T with the minimum congestion achievable at time T. Before we analyze Algorithm A, let us define some useful notation:

- $\lambda_A \equiv \max_{e \in E} f(e)$ achieved by Algorithm A at time T.

- $\lambda^* \equiv$ min ($\max_{e \in E} f(e)$) where the minimum is taken over all possible routings of the (s,t) pairs that arrived by time T.

- $cost(s,t) \equiv$ the cost of the path chosen by Algorithm A to connect pair (s,t). This is the cost of the path AT THE TIME THE PATH WAS CHOSEN (i.e., the cost the algorithm "paid" for the path).

- $dist_C(s,t) \equiv$ the minimum cost of an s-t path AT TIME T.

We will prove the following theorem:

Theorem 1 *If* $\lambda^* \leq 1$ (*i.e., if disjoint paths exist for all* (s,t) *pairs that are routed*), *then* $\lambda_A \leq \log_{3/2}(2m)$.

In order to prove Theorem 1, we make use of two lemmas.

NOTE: Throughout the following proofs, $c(e)$ will denote the cost of edge e AT TIME T, and $f(e)$ will denote the number of paths that Algorithm A has routed on edge e AT TIME T.

Lemma 1 *If* $\lambda^* \leq 1$, *then* $\Sigma_{\text{pairs}(s,t)} dist_C(s,t) \leq \Sigma_{e \in E} c(e)$.

Proof. Let P_{st}^* be the path the optimal solution uses to connect pair (s,t). We have that $\Sigma_{\text{pairs}(s,t)} dist_C(s,t) \leq \Sigma_{\text{pairs}(s,t)} [\Sigma_{e \in Pst*} c(e)]$

(since the rhs intermost sum is the cost of an s-t path at time T, and by definition, $dist_C(s,t)$ is the minimum cost of an s-t path at time T)

$$\leq \Sigma_{e \in E} c(e)$$

(each edge occurs at most once in the double summation since the paths P_{st}^* are disjoint).

■

Lemma 2 $\Sigma_{\text{pairs}(s,t)} cost(s,t) = 2(\Sigma_{e \in E} c(e) - m)$.

Proof.

$\Sigma_{\text{pairs}(s,t)} cost(s,t) = \Sigma_{e \in E} [\Sigma_{\text{pairs}(s,t)}$ [contribution of edge e to $cost(s,t)$]]

$$= \Sigma_{e \in E:\, f(e) \geq 1} \left(1 + (3/2) + (3/2)^2 + (3/2)^3 + \ldots + (3/2)^{f(e)-1}\right)$$

(The first time Algorithm A uses an edge e in a path, the edge has cost $(3/2)^0 = 1$. The second time Algorithm A uses edge e in a path, it has cost $(3/2)^1 = 3/2$. Since at time T the edge e has been used in $f(e)$ paths, the total cost paid by Algorithm A for using edge e in **all** the routed s-t paths is precisely the partial geometric sum inside the parentheses on the rhs.)

$$= \Sigma_{e \in E:\, f(e) \geq 1} \left((3/2)^{f(e)} - 1\right) / \left((3/2) - 1\right)$$

(see your favorite calculus text)

$$= 2 \Sigma_{e \in E} \left(c(e) - 1\right)$$

(by definition of $c(e)$, and since the above holds if we include $e \in E:\, f(e) = 0$)

$$= 2\left(\Sigma_{e \in E}\, c(e) - m\right) \quad \blacksquare$$

We are now ready to prove Theorem 1:

Proof of Theorem 1.

$$\begin{aligned}
\Sigma_{e \in E}\, c(e) &\geq \Sigma_{\text{pairs}(s,t)}\, dist_c(s,t) && \text{(by Lemma 1)} \\
&\geq \Sigma_{\text{pairs}(s,t)}\, cost(s,t) && \text{(since } c \text{ is nondecreasing throughout Algorithm } A\text{)} \\
&= 2\left(\Sigma_{e \in E}\, c(e) - m\right) && \text{(by Lemma 2)}
\end{aligned}$$

$$\begin{aligned}
&\Rightarrow && 2m \geq \Sigma_{e \in E}\, c(e) \\
&\Rightarrow && 2m \geq c(e) \quad \forall\, e \in E \\
&\Rightarrow && 2m \geq (3/2)^{f(e)} \quad \forall\, e \in E \\
&\Rightarrow && \log_{3/2}(2m) \geq f(e) \quad \forall\, e \in E \\
&\Rightarrow && \log_{3/2}(2m) \geq \lambda_A \quad \blacksquare
\end{aligned}$$

Comments

1. Our choice of 3/2 as the base for the cost function in the algorithm was somewhat arbitrary. In particular, for **any** $1 < \alpha < 2$, we can use α as the base for the cost function c and the algorithm remains $\log_\alpha(m / (2-\alpha))$-competitive.

2. Lemma 1 holds for **any** nonnegative cost function c.