CS294-1 On-line Computation & Network Algorithms **Spring 1997**

Lecture 26: May 8

Lecturer: Yair Bartal *Scribe: Anupam Gupta*

In this lecture we study the problem of online virtual circuit routing.

26.1 Virtual Circuit Routing

The problem is the following: we are given a network in which links have finite capacity. We get requests for connections, each of which requests a certain rate and has a certain benefit along with it. We have to decide whether to accept the request, and if so, we have to decide upon a path connecting the nodes.

The result presented here is a simplification of a result by Awerbuch, Azar and Plotkin [AAP93] which is based on techniques of Aspnes et al. [AAFPW92] applied to load balancing on unrelated machines and load minimizatiom in virtual circuit routing, which in themselves are drawn out of various papers dealing with multicommodity flow problems. These methods have also applications for distributed flow computations.

26.1.1 Preliminaries and Definitions

We represent the network as a capacitated graph $G(V, E, c)$, where c_e represents the capacity of the edge e. The request sequence is a collection of connection requests $\beta_i = (s_i, t_i, r_i, B_i, D_i)$, where s_i and t_i are the source and target of the connection, r_i represents the rate requested, B_i is the benefit and D_i is the duration for which the connection must be made. Upon receiving the request, the algorithm either *routes* it by assigning it a path P_i from s_i to t_i, or *rejects* it. For notational convenience, we shall set $P_i = \phi$ if the path is rejected. We can now define the relative load on edge e to be

$$\lambda_e = \sum_{i:P_i \ni e} \frac{r_i}{c_e}.$$

The capacity constraints require that this value be always bounded above by 1.

It is rather difficult to get good bounds for this general problem, so we look at constrained versions of this. Here we restrict the requests so that the rates r_i are small compared to the capacity of the edges. More precisely, we make sure that for all requests β_i and all edges e, $r_i/c_e \le (\log 2mBCR)^{-1}$, where the other terms are defined as under:

$$\begin{aligned} B &= \max_i \{B_i\} \\ C &= \max_e \{c_e\} \\ R &= 1/\min_i \{r_i\} \end{aligned}$$

For brevity, we let $S = 2mBCR$, then our assumption is $r_i/c_e \leq (\log S)^{-1}$.

26.1.2 The Algorithm

The admission control and routing algorithm is very simple. We first define a quantity called the edge cost to be $X_e = 2^{\lambda_e \log S}/m$. Now, given a request β_i, if there is a path P s.t. $\sum_{e \in P}(X_e/c_e) \leq B_i/r_i$, then we accept and let $P_i = P$. Otherwise, we reject.

Lemma 26.1 *The algorithm always maintains capacity constraints.*

Proof: Informally, this is because an edge which is close to being saturated has too high a cost to be ever used.

We show that if there is an edge e such that $\lambda_e > 1 - 1/\log S$, then no path containing e will be accepted. This is because $X_e = 2^{\lambda_e \log S}/m > 2^{S-1} = BCR$. Thus the sum over the path P, $\sum_{\bar{e} \in P} \frac{X_{\bar{e}}}{c_{\bar{e}}} \geq \frac{X_e}{c_e} > BR \geq \frac{B_i}{r_i}$. ■

26.1.3 Performance Bounds

Now we will give the competitive ratio by using the edge costs to bound the optimal and online costs. Let us define $X = \sum_e X_e$ and initial and final values of X be $X(o)$ and $X(f)$. Note that $X(o) = m \cdot \frac{1}{m} = 1$. Further, let the relative load for edges in the offline algorithm's network be λ_e^*. Finally, note that the assumptions ensure that $\Delta\lambda_e \leq 1/\log S$.

Lemma 26.2 *If $B(OPT)$ is the benefit to the optimal algorithm, and $B(ON)$ to the online algorithm, then $X(f) \geq B(OPT) - B(ON)$.*

Proof: Let $\mathcal{Q}$ be the set of indices of the connections that are admitted by the optimal algorithm but not by the online algorithm. Also let P_i' be the path used by the offline algorithm to route the β_i. Then:

$$\begin{aligned}
X(f) &= \sum_e X_e(f) \geq \sum_e \lambda_e^*(f) X_e(f) \\
&= \sum_e \sum_{e \in P_i' \in OPT} \frac{r_i}{c_e} X_e(f) \\
&\geq \sum_{i \in \mathcal{Q}} r_i \sum_{e \in P_i'} \frac{X_e(f)}{c_e} > \sum_{i \in \mathcal{Q}} r_i \cdot \frac{B_i}{r_i} = \sum_{i \in \mathcal{Q}} B_i \\
&\geq B(OPT) - B(ON).
\end{aligned}$$

■

Lemma 26.3 *For every request, $\Delta X \leq \log S \cdot \Delta B(ON)$.*

Proof: This lemma tries to lower bound the benefit accrued by our algorithm.

$$\begin{aligned}
\Delta X &= \sum_e \Delta X_e = \sum \left(\frac{2^{(\lambda_e + \Delta\lambda_e)\log S}}{m} - \frac{2^{\lambda_e \log S}}{m} \right) \\
&= \sum_e X_e (2^{\Delta\lambda_e \log S} - 1) \\
&\le \sum_e X_e (\Delta\lambda_e \log S - 1) \qquad ((2^x - 1) \le x \text{ if } x \in [0,1]) \\
&\le \sum_{e \in P_i} X_e \frac{r_i}{c_e} \log S \\
&\le B_i \log S \qquad (\textstyle\sum_{e \in P_i} \frac{X_e}{c_e} \le \frac{B_i}{r_i}) \\
&= \Delta B(ON) \log S.
\end{aligned}$$

■

Theorem 26.4 *The algorithm has a competitive ratio of* $\log S$.

Proof: From lemma 26.2, we have that $X(f) - X(o) \ge B(OPT) - B(ON) - 1$. And from lemma 26.3, we have that $X(f) - X(o) \le B(ON) \cdot \log S$. Hence we get that

$$\begin{aligned}
B(OPT) &\le B(ON)(\log S + 1) + 1 \\
&\le B(ON)(\log S + 2).
\end{aligned}$$

■

References

[AAFPW92] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin and O. Waarts. On-line Load Balancing with Applications to Machine Scheduling and Virtual Circuit Routing. In *Proc. of the 33rd Ann. IEEE Symp. on Foundations of Computer Science*, pages 164-173, October 1992.

[AAP93] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput Competitive On-line Routing. In *Proc. of the 34th Ann. IEEE Symp. on Foundations of Computer Science*, November 1993.