GREEDY ON-LINE FILE TRANSFER ROUTING[*]

JESSEN T. HAVILL AND WEIZHEN MAO
Department of Computer Science, The College of William and Mary
Williamsburg, Virginia, USA 23187-8795

ABSTRACT

We study the performance of a simple greedy on-line algorithm for routing large file transfers in a network. The goal of the algorithm is to minimize network congestion. We show that the competitive ratio of the greedy algorithm is $\mathrm{O}\left(L \min\left\{\frac{\Lambda}{\lambda}, I\right\} \log(nI)\right)$ where L is the length of the longest path assigned to a file transfer, $\frac{\Lambda}{\lambda}$ is the ratio of the maximum to minimum file length, I is a lower bound on the overlap of optimal paths, and n is the number of nodes in the network. We also show that this upper bound is close to being tight when L is small by proving that the competitive ratio is $\Omega\left(L + \log\left(\frac{n}{L} - L\right)\right)$.

Keywords: Network routing, On-line algorithms, Competitive analysis, Greedy algorithms

1 INTRODUCTION

We study a simple greedy algorithm for assigning source routes to large file transfers in a network. Whereas in traditional packet switching, the cost of computing such a route globally for each packet is much too prohibitive to be practical, this idea seems feasible in the context of large files. In such cases, the time required to globally determine a good route for a file transfer is likely to be small compared to the time required for the actual transfer. Furthermore, transferring files efficiently in light of current traffic conditions is more important for large files than for small ones due to longer delays arising from collisions. The traditional packet routing method cannot (for lack of global information) prevent collisions between files, whereas a centralized method can carefully spread out file transfers to reduce network congestion. This problem is becoming quite relevant as more scientific and multimedia applications develop that require the transfer of large amounts of information. Network backups and file redistribution in distributed databases [1] are also promising candidates.

In order to reflect many real situations, we study the *on-line* version of the problem in which file transfer requests become known to an algorithm one at a time and the routing decision must be made for each request before any future requests are known. This approach is in contrast to the traditional *off-line* approach in which it is assumed that all requests are known *a priori*. In order to measure the efficiency of an on-line algorithm, we compare its performance, with respect to some objective function, to that of an optimal off-line algorithm, a technique known as *competitive analysis* [2]. In contrast to worst case analysis, this method takes into account the inherent difficulty of an instance and asks the algorithm to perform well only with respect to this difficulty.

Our problem originated from the File Transfer Scheduling problem of Coffman, Garey, Johnson, and LaPaugh [3], which did not involve routing; rather the object was to schedule the transfer of files over direct connections while obeying port constraints at the network nodes. Subsequently, the off-line version of our problem was considered independently by two groups of authors. Rivera-Vega, Varadarajan and Navathe [4] consider the special case on a general network graph with unit speed links, infinite capacity buffers and unit file sizes, and show it is NP-hard in general. In their model, links are switched on a file by file basis and the goal is to minimize makespan. Rivera-Vega, Varadarajan and Navathe [1] and Varadarajan and Rivera-Vega [5] study the same version of the problem on fully connected networks. Mao and Simha [6, 7] later proved that the problem is NP-hard even on very simple graphs and also studied some list scheduling heuristics. The routing component of our problem is a special case of the on-line virtual circuit routing problem studied by Aspnes, Azar, Fiat, Plotkin, and Waarts [8]. These authors' algorithm for the problem, which bases each routing decision on the computation of an exponential function of network congestion, gives $\mathrm{O}(\log n)$ (where n is the number of network nodes) competitive solutions on arbitrary networks even if the load incurred on a link is not related to the link's speed. Their general problem is a generalization of the on-line load balancing problem on unrelated machines, for which the natural greedy algorithm has been shown to have a linear competitive ratio. In contrast, our special case of virtual circuit routing is *not* a generalization of on-line load balancing on unrelated machines, as the load incurred on an edge in our case is always the file length divided by the link speed. However, our problem *is* a generalization of the on-line load balancing problem with assignment restriction, for which Azar, Naor, and Rom [9] show the greedy algorithm is be-

[*]This research is supported in part by NSF grant NCR-9505963.

tween $\lceil \log(n+1) \rceil$ and $\lceil \log n \rceil + 1$ competitive, where n is the number of machines. One of our goals is to learn whether this greedy algorithm can be extended to routing, and still give a small competitive ratio.

Our general problem of interest is the On-line File Transfer Routing and Scheduling problem, formally defined as follows. We are given a weighted directed graph $G = (V, E, s, m)$ representing the topology of a communication network. The set V of vertices represents a set of machines which are capable of sending and receiving files. The set E of directed edges represents a set of communication links between machines. The function $s : E \to \{1, 2, 3, \ldots\}$ describes the *speed* of the network links (i.e., the number of packets that can be transmitted over edge e in one time unit). In this paper, we assume that $s(e) = 1$ for all $e \in E$. The function $m : V \to \{0, 1, 2, \ldots\}$ describes the *memory constraints* associated with the network nodes. In particular, $m(v)$ is the size of the buffer space available at machine v for forwarding packets.

The input to our problem is a sequence of file transfer requests $\sigma = f_1, f_2, \ldots, f_k$. Each $f_j \in \sigma$ is represented by a quadruple (s_j, t_j, l_j, a_j). The nodes $s_j, t_j \in V$ are the *source* and *destination* of the file transfer, respectively. The value $l_j \in \{1, 2, 3, \ldots\}$ is the *length* of the file (in fixed-size packets), and $a_j \in \{0, 1, 2, \ldots\}$ is the request's *arrival time*. In general, when request f_j arrives, an algorithm must immediately make both a *routing* and *scheduling* decision based solely on the decisions it made for requests $f_1, f_2, \ldots, f_{j-1}$. An algorithm routes a file transfer f_j by assigning to f_j a path $P_j \in \mathcal{P}_j$, where $\mathcal{P}_j$ is the set of paths between machines s_j and t_j in the network. (We assume that $|\mathcal{P}_j| > 0$ for all $j = 1, 2, \ldots, k$.) An algorithm also schedules the file transfer by specifying the times at which it will cross each link in P_j. If a file transfer is not crossing a link and has not yet reached its destination, then it must be assigned to a buffer at the head of the next link on its path, but no node v may have more than $m(v)$ packets assigned to it at any time.

The objective function we consider is the network *congestion*, defined to be the maximum ratio, over all links in the network, of the number of packets that crossed the link to the link's speed (which in this paper is 1). Formally, we define

$$\mu_j(e) = \sum_{i \le j : e \in P_i} l_i$$

to be the *congestion on edge e* after an on-line algorithm has routed requests $f_1, f_2, \ldots, f_j$. For simplicity, let $\mu(e) = \mu_k(e)$. The *congestion on path P* after an on-line algorithm has routed requests $f_1, f_2, \ldots, f_j$ is defined to be $\mu_j(P) = \max_{e \in P} \mu_j(e)$. Let $\mu(P) = \mu_k(P)$. Finally, the *network congestion* after an on-line algorithm has routed requests $f_1, f_2, \ldots, f_j$ is defined to be $\mu_j = \max_{e \in E} \mu_j(e)$. The total congestion incurred by an on-line algorithm is denoted $\mu = \mu_k$. Note that the network congestion created by an algorithm is independent of any scheduling decisions. Thus, in this paper, we ignore the scheduling component of the problem (and hence the memory constraints as well).

We study a simple routing algorithm called Greedy, which is defined as follows:

Algorithm Greedy: For a request f_j, assign any route $P \in \mathcal{P}_j$ such that $\mu_{j-1}(P) = \min_{P' \in \mathcal{P}_j} \mu_{j-1}(P')$.

In other words, Greedy assigns the route that will have the minimum congestion among all possible paths for that file transfer. Greedy is attractive because of its speed and simplicity of implementation. Whereas each routing decision of the $O(\log n)$ competitive algorithm of [8] requires several computationally expensive calculations of an exponential function of the congestion, this algorithm simply computes the maximum of a set of integers.

As mentioned earlier, we use *competitive analysis* to analyze the performance of an on-line algorithm. Formally, consider an on-line algorithm A for a minimization problem. Let $C_A(\sigma)$ be the cost of the solution obtained by A given the request sequence σ. Let $C_{OPT}(\sigma)$ be the cost obtained by an optimal off-line algorithm when given σ. We say that A is *c-competitive* if and only if, for all request sequences σ, $C_A(\sigma) \le c \cdot C_{OPT}(\sigma) + a$, where a is a constant. The *competitive ratio* of A is

$$c_A = \sup_{\sigma} \frac{C_A(\sigma)}{C_{OPT}(\sigma)}.$$

Havill, Mao, and Simha [10] proved that the competitive ratio of any on-line algorithm for our problem is at least $\left\lceil \frac{\log n + 2}{2} \right\rceil$ with respect to congestion, where n is the number of nodes in the network. The lower bound holds even when all files have unit length and share one destination, and the network is a simple 2-layer graph with unit-speed links.

The remainder of the paper is outlined as follows. In Section 2, we prove that the competitive ratio of Greedy is $O\left(L \min\left\{\frac{\Lambda}{\lambda}, I\right\} \log(nI)\right)$, where L is the length of the longest path assigned to a file transfer, $\frac{\Lambda}{\lambda}$ is the ratio of the maximum to minimum file length, I is a lower bound on the overlap of optimal paths (which we will define later), and n is the number of nodes in the network. If $I = O(1)$, or if $\frac{\Lambda}{\lambda} = O(1)$ and $I = n^{O(1)}$, our result shows that the competitive ratio is $O(L \log n)$. Furthermore, if $L = O(\log n)$, as is the case in some common networks, the competitive ratio of Greedy is polylogarithmic in n. In Section 3, we show that this bound is close to being tight when L is small by proving that the competitive ratio of the greedy algorithm is $\Omega\left(L + \log\left(\frac{n}{L} - L\right)\right)$. Note that if $L = O(1)$ and either of the previous conditions on I

and $\frac{\Lambda}{\lambda}$ is also true, then GREEDY is $\Theta(\log n)$ competitive. Finally, in Section 4 we draw some conclusions and summarize our results.

2 AN UPPER BOUND

In this section, we prove an upper bound on the competitive ratio of GREEDY. The method we use to prove our result is adapted from a technique used by Azar, Naor and Rom [9] to show that an algorithm similar to GREEDY is $O(\log n)$ competitive for an on-line load balancing problem in which each job can be assigned to exactly one of a subset of n identical machines. Intuitively, their idea was to conceptualize the on-line algorithm's assignment on each machine as a partition of a number of successive *layers*. The sum of the weights of the jobs in each layer (which we will call the layer's *width*) is equal to the optimal load, except for possibly the last layer which contains the remaining weight less than the optimal load. (All subsequent layers after this last nonzero one have width 0.) They show that, for any i, the sum of the widths of all the i^{th} layers is at least as large as the sum of the widths of all subsequent layers. From this step, the logarithmic competitive ratio follows in a relatively straightforward way. In our proof, we adapt this method to work with paths, rather than single machines. Our idea is to consider the final on-line congestion caused by GREEDY on each *optimal path* and partition the congestion on each of these optimal paths into layers with width at most Λ, the maximum file length. We then show that, for any i, the sum of the widths of all the i^{th} layers, multiplied by a certain factor, is at least as large as the sum of the widths of all subsequent layers, and our result follows. Before formally stating our theorem and its proof, we will present the notation we will use to formally describe these ideas, along with several useful facts.

- P_j^* is the path chosen for file transfer f_j by an optimal algorithm.
- μ^* refers to the network congestion incurred by an optimal algorithm.
- $L = \max_j \max_{P \in \mathcal{P}_j} |P|$ is the length of the longest path that can be assigned to a file transfer.
- $\lambda = \min_j l_j$ and $\Lambda = \max_j l_j$ are the minimum and maximum file lengths, respectively.

  Fact 1 $\mu^* \geq \Lambda$.

  Fact 2 $\mu^* \geq \frac{1}{m}\sum_{j=1}^{k} l_j$.

- $e_j^* \in P_j^*$ is an edge $e \in P_j^*$ satisfying $\mu(e) = \mu(P_j^*)$.
- $I = \max_{e \in E} \left|\left\{j : e_j^* = e, 1 \leq j \leq k\right\}\right|$ is the maximum number of optimal paths P_j^* whose edge e_j^* is the same. The following fact is true since I is a lower bound on the maximum number of optimal paths that intersect at the same edge.

  Fact 3 $\mu^* \geq I\lambda$.

- $W_{ij} = \begin{cases} \Lambda, & \mu(P_j^*) \geq i\Lambda \\ \mu(P_j^*) - (i-1)\Lambda, & (i-1)\Lambda < \mu(P_j^*) < i\Lambda \\ 0, & \text{otherwise} \end{cases}$

  As discussed above, we partition the congestion on edge e_j^* into layers with width at most Λ. W_{ij} is the width of the i^{th} such layer.

  Fact 4 $\mu(P_j^*) = \sum_i W_{ij}$.

- W_{ij}^r is the portion of W_{ij} on edge e_j^* that was added by file transfer f_r in the on-line route assignment. The following fact is true because if $W_{ij}^r > 0$ for more than 2 layers, then $l_r > \Lambda$, which is impossible.

  Fact 5 *$W_{ij}^r > 0$ for at most 2 values of i, which must be consecutive.*

- $S_{ij} = \{r : W_{lr}^j > 0 \text{ for some } l > i\}$ is the set of indices of optimal paths P_r^* such that f_j traverses e_r^* in a layer greater than i in the on-line route assignment. The following fact is true because f_j may traverse L edges, each of which may be e_r^* for at most I optimal paths P_r^*.

  Fact 6 *For all i and j, $|S_{ij}| \leq LI$.*

- $R_{ij} = \sum_{r \in S_{ij}} (l_j - W_{ir}^j)$ is the total congestion incurred by f_j in all layers greater than i. The equation is correct because if $r \in S_{ij}$, then $W_{lr}^j = 0$ for all $l < i$, by Fact 5.

  Fact 7 *For all i and j, $R_{ij} \leq |S_{ij}| \cdot l_j \leq LI\Lambda$.*

- $W_i = \sum_{j=1}^{k} W_{ij}$.
- $R_i = \sum_{j=1}^{k} R_{ij}$.

  Fact 8 $R_i = \sum_{l>i} W_l$.

  Proof Notice that

$$\begin{aligned} R_i &= \sum_{j=1}^{k} \sum_{r \in S_{ij}} \left(l_j - W_{ir}^j\right) \\ &= \sum_{r=1}^{k} \sum_{j : r \in S_{ij}} \left(l_j - W_{ir}^j\right) \\ &= \sum_{r=1}^{k} \sum_{l>i} W_{lr} \\ &= \sum_{l>i} W_l. \end{aligned}$$

An explanation is in order for the third equality. The term $\sum_{j:r \in S_{ij}} (l_j - W_{ir}^j)$ refers to the total length of all file transfers f_j that cross e_r^* after layer i, minus the portions that fall in layer i. (No portion may fall in a layer before i by Fact 5.) This is simply the total width of all the layers of e_r^* after layer i, or $\sum_{l>i} W_{lr}$. ■

The next fact follows from Fact 8.

Fact 9 $R_i = R_{i-1} - W_i$.

Fact 10 $R_0 \le LI \sum_{j=1}^{k} l_j$.

We can now formally state our theorem:

Theorem 1 *The competitive ratio of* Greedy *with respect to congestion is* $\mathrm{O}\left(L \min\left\{\frac{\Lambda}{\lambda}, I\right\} \log(nI)\right)$.

We will use the next three lemmas in our proof.

Lemma 1 $\mu \le \max_j \mu(P_j^*) + \Lambda$. (Proof omitted.)

Lemma 2 *For all i and j, $LI \cdot W_{ij} \ge R_{ij}$.*

Proof We divide the proof into three cases. The lemma follows trivially in the first two cases; the main part of the proof is contained in the third case.

Case 1: $R_{ij} = 0$. In this case, the lemma is clearly true since the LHS is always non-negative.

Case 2: $W_{ij} = \Lambda$. In this case, the lemma is also clearly true since, by Fact 7, $R_{ij} \le LI\Lambda$.

Case 3: $R_{ij} > 0$ and $W_{ij} < \Lambda$. Since $R_{ij} > 0$, we know, by definition, that $S_{ij} \ne \emptyset$. Let r be an arbitrary member of S_{ij}. Then, by the definition of S_{ij}, for some $l > i$, $W_{lr}^j > 0$. This means that f_j crossed edge e_r^* and therefore $e_r^* \in P_j$. Now notice that since $W_{ij} < \Lambda$, the i^{th} layer of e_j^* is the last nonzero layer, and thus $\mu(P_j^*) < i\Lambda$ by the definition of W_{ij}. So clearly, $\mu_{j-1}(P_j^*) < i\Lambda$ and, since $\mu_{j-1}(P_j) \le \mu_{j-1}(P_j^*)$ by the definition of Greedy and $e_r^* \in P_j$, we know that

$$\mu_{j-1}(e_r^*) < i\Lambda. \tag{1}$$

Also, notice that, since $W_{lr}^j > 0$ for some $l > i$,

$$\mu_j(e_r^*) > i\Lambda. \tag{2}$$

In other words, (1) and (2) tell us that just before f_j was assigned to route P_j, the edge $e_r^* \in P_j$ had congestion less than $i\Lambda$ (but greater than $(i-1)\Lambda$ since $l_j \le \Lambda$), and just after f_j was assigned, e_r^* had congestion greater than $i\Lambda$. Thus, the value l_j added to the congestion on edge e_r^* is divided between layers i and $i+1$: W_{ir}^j is added to layer i and $W_{(i+1)r}^j$ is added to layer $i+1$. Thus,

$$\begin{aligned}\mu_{j-1}(P_j) &\ge \mu_{j-1}(e_r^*)\\ &= (i-1)\Lambda + (\Lambda - W_{ir}^j).\end{aligned} \tag{3}$$

We next need to show that

$$W_{(i-1)j} = \Lambda. \tag{4}$$

Assume, for contradiction, that $W_{(i-1)j} < \Lambda$. This implies that $\mu(P_j^*) < (i-1)\Lambda$, and therefore $\mu_{j-1}(P_j^*) < (i-1)\Lambda$. But this means that f_j should not have been assigned to P_j since $\mu_{j-1}(P_j) > (i-1)\Lambda$. Thus, (4) is true. Now, by combining (4) with the assumption that $W_{ij} < \Lambda$, we see that

$$\mu(P_j^*) = (i-1)\Lambda + W_{ij}. \tag{5}$$

From (5), the definition of Greedy, and (3), we can deduce the following:

$$\begin{aligned}(i-1)\Lambda + W_{ij} &= \mu(P_j^*)\\ &\ge \mu_{j-1}(P_j)\\ &\ge (i-1)\Lambda + (\Lambda - W_{ir}^j).\end{aligned}$$

Thus,

$$W_{ij} \ge \Lambda - W_{ir}^j. \tag{6}$$

Finally, we can conclude that

$$\begin{aligned}R_{ij} &= \sum_{r \in S_{ij}} \left(l_j - W_{ir}^j\right) && \text{by the def. of } R_{ij}\\ &\le |S_{ij}| \left(\Lambda - \min_{r' \in S_{ij}} W_{ir'}^j\right)\\ &\le LI \cdot W_{ij} && \text{by Fact 6 and (6).}\end{aligned}$$

■

Lemma 3 *For any i, $R_i \le \left(\frac{LI}{LI+1}\right)^i R_0$.*

Proof By Lemma 2 and the definitions of W_i and R_i, we know that

$$W_i = \sum_{j=1}^{k} W_{ij} \ge \frac{1}{LI} \sum_{j=1}^{k} R_{ij} = \frac{1}{LI}\, R_i. \tag{7}$$

Thus, from (7) and Fact 9, $R_i \le R_{i-1} - \frac{1}{LI}\, R_i$. So,

$$R_i \;\le\; \frac{LI}{LI+1}\, R_{i-1}. \tag{8}$$

Finally, by recursively applying (8), we conclude that

$$R_i \;\le\; \left(\frac{LI}{LI+1}\right)^i R_0.$$ ■

Proof of Theorem 1 Let

$$b = \left\lceil \log_{\left(\frac{LI+1}{LI}\right)}(Im) \right\rceil. \tag{9}$$

Then we can deduce that

$$\begin{aligned}R_b &\le \left(\frac{LI}{LI+1}\right)^b R_0 && \text{by Lemma 3}\\ &\le \frac{1}{Im}\, LI \sum_{j=1}^{k} l_j && \text{by (9) and Fact 10}\\ &\le L\mu^* && \text{by Fact 2.}\end{aligned} \tag{10}$$

Now notice that, by Fact 4, Fact 8, the definition of W_{ij}, and (10), for all j,

$$\mu(P_j^*) = \sum_i W_{ij} \le \sum_{i=1}^{b} W_{ij} + R_b \le b\Lambda + L\mu^*.$$

Thus,

$$\max_j \mu(P_j^*) \le b\Lambda + L\mu^* = \left\lceil \log_{\left(\frac{LI+1}{LI}\right)}(Im) \right\rceil \Lambda + L\mu^*.$$

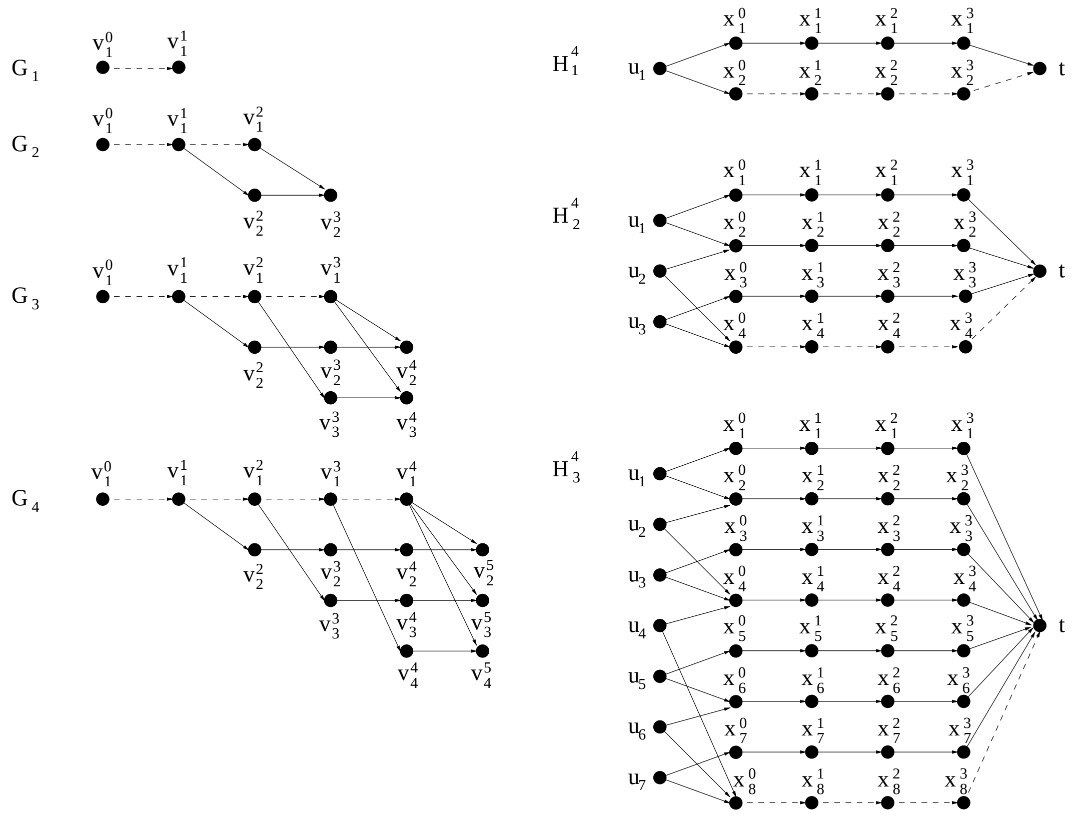

FIGURE 1: SOME EXAMPLES OF G_l AND H^l_j.

By Lemma 1, this implies that

$$\mu \le \left\lceil \log_{\left(\frac{LI+1}{LI}\right)}(Im) \right\rceil \Lambda + \Lambda + L\mu^*.$$

So the competitive ratio of Greedy is

$$\begin{aligned}
\frac{\mu}{\mu^*} &\le \frac{\left\lceil \log_{\left(\frac{LI+1}{LI}\right)}(Im) \right\rceil \Lambda + \Lambda}{\max\{I\lambda, \Lambda\}} + L \\
&\le \frac{LI\Lambda \log_2(Im) + 2\Lambda}{\max\{I\lambda, \Lambda\}} + L \\
&\le \min\left\{ L\frac{\Lambda}{\lambda}\log_2(Im) + 2, LI\log_2(Im) + 2 \right\} + L \\
&= \mathrm{O}\left(L \min\left\{ \frac{\Lambda}{\lambda}, I \right\} \log(nI) \right). \qquad (11)
\end{aligned}$$

The first inequality follows from Facts 1 and 3, and the second inequality follows from the fact that, for any $a, y \ge 1$, $\left\lceil \log_{\left(\frac{a+1}{a}\right)} y \right\rceil \le a \log_2 y + 1$. ■

3 A LOWER BOUND

In this section, we present a lower bound on the competitive ratio of Greedy to complement the upper bound proven in the last section.

Theorem 2 *The competitive ratio of* Greedy *with respect to congestion is* $\Omega\left(L + \log\left(\frac{n}{L} - L\right)\right)$.

Proof To prove the lower bound, we construct a directed network G^l_j (with unit speed links and infinite capacity buffers) which can cause Greedy to incur congestion equal to $l + j$. We begin by defining two auxiliary digraphs, G_l and H^l_j, from which we will construct the final network graph. The digraph G_l, for $l = 1, 2, \ldots$, is defined as follows:

$$\begin{aligned}
V(G_l) &= \left\{v^0_1\right\} \cup \left\{v^h_i \colon 1 \le h \le l,\ 1 \le i \le h\right\} \\
&\quad \cup \left\{v^{l+1}_i \colon i = 2, 3, \ldots, l\right\} \\
E(G_l) &= \left\{\left(v^h_1, v^{h+1}_1\right) \colon 0 \le h \le l-1\right\} \\
&\quad \cup \left\{\left(v^h_1, v^{h+1}_{h+1}\right) \colon 1 \le h \le l-1\right\} \\
&\quad \cup \left\{\left(v^h_i, v^{h+1}_i\right) \colon 2 \le i \le l, i \le h \le l\right\} \\
&\quad \cup \left\{\left(v^l_1, v^{l+1}_i\right) \colon 2 \le i \le l\right\}
\end{aligned}$$

Notice that $|V(G_l)| = \frac{l(l+3)}{2}$. The digraph H^l_j, for $l = 1, 2, \ldots$ and $j = 1, 2, \ldots$, is defined as follows:

$$\begin{aligned}
V\left(H^l_j\right) &= \left\{u_i \colon 1 \le i \le 2^j - 1\right\} \\
&\quad \cup \left\{x^h_i \colon 1 \le i \le 2^j,\ 0 \le h \le l-1\right\} \cup \{t\} \\
E\left(H^l_j\right) &= \left\{\left(u_i, x^0_i\right), \left(u_i, x^0_{i+1}\right) \colon i \text{ odd}, 1 \le i \le 2^j - 1\right\} \\
&\quad \cup \left\{\left(u_i, x^0_i\right), \left(u_i, x^0_{i+2^{j-m}}\right) \colon i \text{ even}, \right. \\
&\qquad \left. 2 \le i \le 2^j - 2, m = (|2^{j-1} - i|)/2 + 1\right\} \\
&\quad \cup \left\{\left(x^h_i, x^{h+1}_i\right) \colon 1 \le i \le 2^j,\ 0 \le h \le l-2\right\} \\
&\quad \cup \left\{\left(x^{l-1}_i, t\right) \colon 1 \le i \le 2^j\right\}
\end{aligned}$$

Notice that $|V(H^l_j)| = (l+1)2^j$. Examples of G_l and H^l_j are shown in Figure 1. The digraph G^l_j is constructed by combining graphs G_l and H^l_j in the following way: First, in graph H^l_j, for each $h = 0, 1, \ldots, l-1$, rename node $x^h_{2^j}$ to be v^h_1. Also rename node t to be

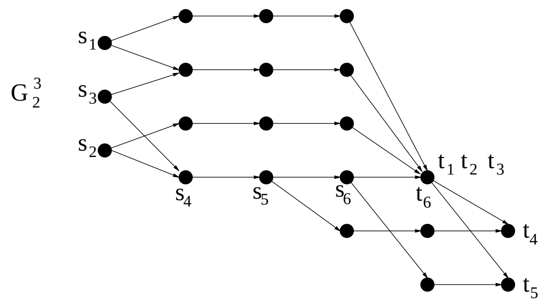

FIGURE 2: GRAPH G_2^3.

v_1^l. Call this new graph $\hat{H}_j^l$. Second, let $G_j^l = \hat{H}_j^l \cup G_l$. In other words, to construct G_j^l we combine the two graphs G_l and H_j^l so that the dashed edges in Figure 1 overlap. As an example, G_2^3 is shown in Figure 2. Notice that G_j^l has $n = |V(G_l)| + |V(H_j^l)| - (l+1)$ vertices.

In order to define the request sequence for G_j^l, we first define request sequences for the graphs G_l and H_j^l. Each file transfer f_i has length 1 and arrival time i, and so we denote each request simply by (s_i, t_i). The request sequence $\sigma(G_l)$ for G_l consists of the l requests defined as follows:

$$f_i = \begin{cases} \left(v_1^{i-1}, v_{i+1}^{l+1}\right), & i = 1, 2, \ldots, l-1 \\ \left(v_1^{l-1}, v_1^l\right), & i = l \end{cases}$$

The request sequence $\sigma(H_j^l)$ for H_j^l consists of $2^j - 1$ requests organized into j phases. In phase i, where $1 \le i \le j$, we issue the 2^{j-i} requests $(u_{2^{i-1}+2^i \cdot m}, t)$, for $m = 0, 1, \ldots, 2^{j-i} - 1$. For example, for any H_3^l, 7 requests are made. Phase 1 contains requests (u_1, t), (u_3, t), (u_5, t), and (u_7, t). Phase 2 contains requests (u_2, t) and (u_6, t). Phase 3 contains request (u_4, t). Finally, request sequence σ for G_j^l consists of the request sequence for H_j^l followed by the request sequence for G_l, with vertices relabeled appropriately. Figure 2 shows the sources and destinations of the requests for G_2^3.

If Greedy chooses poorly, it can incur a congestion of $l + j > (L-1) + \log\left(\frac{n+1}{L} - \frac{L}{2}\right)$, where L is the length of the longest path in G_j^l. On the other hand, an optimal algorithm can always find a set of edge disjoint routes with congestion equal to 1. Therefore the competitive ratio for Greedy is at least $\Omega(L + \log(\frac{n}{L} - L))$. As an example, consider again the graph in Figure 2. Greedy could assign requests f_1 and f_2 to their respective "bottom" routes (as drawn in the figure). When f_3 arrives, both of its routes have congestion 1, so it could also choose its bottom route. When f_4 arrives, both routes share an edge with congestion 2, so it might choose its "top" route, causing the first three edges on that route to have congestion 3. Then when f_5 arrives, it could choose its top route as well, causing the first 2 edges on that route to have congestion 4. Finally, f_6 has only one route, whose sole edge will have congestion 5. ■

4 CONCLUSIONS

We have given some results about the performance of an on-line greedy algorithm for routing large file transfers on a network. Although greedy algorithms are not always the best available, it is important to understand something about their efficiency due to their attractive simplicity and ease of implementation. We have shown that the competitive ratio of Greedy with respect to network congestion is at least $\Omega\left(L + \log\left(\frac{n}{L} - L\right)\right)$, but at most $\mathrm{O}\left(L \min\left\{\frac{\Lambda}{\lambda}, I\right\} \log(nI)\right)$. In some reasonable circumstances, we can infer even better results. For example, if the instance is such that $I = \mathrm{O}(1)$, or $\frac{\Lambda}{\lambda} = \mathrm{O}(1)$ and $I = n^{\mathrm{O}(1)}$, then Greedy is $\mathrm{O}(L \log n)$ competitive. On some common networks, $L = \mathrm{O}(\log n)$, which implies a polylogarithmic competitive ratio. Furthermore, if $L = \mathrm{O}(1)$, the competitive ratio is $\Theta(\log n)$.

REFERENCES

[1] P. I. Rivera-Vega, R. Varadarajan, and S. B. Navathe, Scheduling file transfers in fully connected networks, *Networks*, 22, 1992, 563–588.

[2] D. D. Sleator and R. E. Tarjan, Amortized efficiency of list update and paging rules, *Comm. of the ACM*, 28(2), 1985, 202–208.

[3] E. G. Coffman, M. R. Garey, D. S. Johnson, and A. S. LaPaugh, Scheduling file transfers, *SIAM J. on Computing*, 14, 1985, 744–780.

[4] P. I. Rivera-Vega, R. Varadarajan, and S. B. Navathe, Scheduling data redistribution in distributed databases, In *Proc. of the 6th Int. Conf. on Data Engineering*, 1990, 166–173.

[5] R. Varadarajan and P. I. Rivera-Vega, An efficient approximation algorithm for the file redistribution scheduling problem in fully connected networks, TR 92-020, Univ. of Florida, 1992.

[6] W. Mao, R. Simha, and D. Nicol, A general file transfer scheduling problem, *TIMS/ORSA Joint Conference*, Chicago, IL, 1993.

[7] W. Mao and R. Simha, Routing and scheduling file transfers in packet-switched networks, *J. of Computing and Information*, 1, 1994, 559–574.

[8] 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 ACM STOC*, 1993, 623–631.

[9] Y. Azar, J. Naor, and R. Rom, The competitiveness of on-line assignments, *J. of Algorithms*, 18, 1995, 221–237.

[10] J. T. Havill, W. Mao, and R. Simha, A lower bound for on-line file transfer routing and scheduling, *Conf. on Information Sciences and Systems*, Baltimore, MD, 1997.