

Greedy Online Algorithms for Routing Permanent Virtual Circuits

Jessen T. Havill,¹ Weizhen Mao²

¹ Department of Mathematics and Computer Science, Denison University,
Granville, Ohio 43023

² Department of Computer Science, The College of William and Mary,
Williamsburg, Virginia 23187-8795

Received 14 August 1998; accepted 8 October 1998

Abstract: We analyze the competitive ratio of two greedy online algorithms for routing permanent virtual circuits in a network with arbitrary topology and uniform capacity links. We show that the competitive ratio of the first algorithm, with respect to network congestion, is in $\Omega(\sqrt{\mathcal{D}m})$ and $O(\sqrt{\mathcal{D}\mathcal{L}m})$, where m is the number of links in the network, \mathcal{D} is the maximum ratio, over all requests, of the length of the longest path for the request to the length of the shortest path for the request, and \mathcal{L} is the ratio of the maximum-to-minimum bandwidth requirement. We show that the competitive ratio of the second greedy algorithm is in $\Omega(d + \log(n - d))$ and $\min\{O(d \log n), O(\sqrt{\mathcal{D}\mathcal{L}m})\}$ when the optimal route assignment is pairwise edge disjoint, where n is the number of network nodes and d is the length of the longest path that can be assigned to a request. It is known that the optimal competitive ratio for this problem is $\Theta(\log n)$. Aspnes et al. designed a $\Theta(\log n)$ competitive online algorithm that computes an exponential function of current congestion to make each decision. The greedy online algorithms, although not optimal, make each decision more quickly and still have good competitive ratios in many nontrivial situations. © 1999 John Wiley & Sons, Inc. Networks 34: 136–153, 1999

1. INTRODUCTION

The problem of assigning routes to virtual circuit requests in B-ISDN networks has become an important topic of research. Each virtual circuit request consists of a bandwidth requirement between two nodes in the network, to support some service requiring a quality of service (QOS) guarantee. It is likely that such requests will arrive *online*: An

algorithm will need to assign a route to each request before future requests (if any) are known. When nothing is known about the arrival process of the requests, it is customary to analyze online algorithms with respect to their *competitive ratio*, the supremum over all request sequences of the ratio between the performance of the online algorithm, and that of an *offline* algorithm which can see the future and use this knowledge to make its decisions optimally.

We are interested in the problem of routing permanent virtual circuit requests, those that are assumed to exist indefinitely. It is known that the optimal competitive ratio for this problem is $\Theta(\log n)$ with respect to network con-

Correspondence to: J. T. Havill; e-mail: havill@denison.edu
Contract grant sponsor: NSF; contract grant number: NCR-9505963

gestion, where n is the number of network nodes. Aspnes et al. [1, 2] designed an optimal $O(\log n)$ competitive online algorithm that computes an exponential function of current congestion to make each decision. We will study the competitive ratio of two simpler greedy online algorithms for networks with arbitrary topology and uniform capacity links that, although not optimal, make each decision much more quickly and still have good competitive ratios for many nontrivial situations.

1.1. Problem Statement

The online permanent virtual circuit routing problem is formally defined as follows: We are given a weighted directed graph $G = (V, E, c)$ representing a communication network. The set V represents a set of machines and the set E represents a set of unidirectional communication links between machines. The function $c : E \rightarrow \{1, 2, 3, \dots\}$ describes the capacity of the network links. We will assume that, for all $e \in E$, $c(e)$ is equal to an arbitrary constant w . We do not believe that such an assumption causes much loss of practical generality, especially in the context of backbone networks. The input to our problem is a sequence of permanent virtual circuit requests $\sigma = f_1, f_2, \dots, f_k$. Each $f_j \in \sigma$ is represented by a tuple (s_j, t_j, l_j, a_j) . The nodes $s_j, t_j \in V$ are the source and destination of the request, respectively. The value $l_j \in \{1, 2, 3, \dots\}$ is the required bandwidth of the virtual circuit, and $a_j \in \{0, 1, 2, \dots\}$ is the request's arrival time. Each permanent virtual circuit is assumed to exist indefinitely. Equivalently, one may think of this model as one in which the finishing times of all requests are the same. When a request $f_j \in \sigma$ arrives, a centralized online algorithm must immediately assign 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, based solely on the decisions it made for requests f_1, f_2, \dots, f_{j-1} .

We will seek to minimize the *network congestion*, defined to be the maximum, over all links in the network, of the ratio of the bandwidth assigned to the link to the link's capacity. Minimizing congestion attempts to avoid hot spots or bottlenecks by spreading the bandwidth as evenly as possible over the network links. Minimizing congestion also attempts to increase the network bandwidth available for other users and applications. Formally, we define

$$\mu_j(e) = \sum_{i < j; e \in P_i} \frac{l_i}{c(e)} \quad (1)$$

to be the *congestion on link e* after an online algorithm has routed requests f_1, f_2, \dots, f_{j-1} . (The particular online algorithm will always be clear in the current context.) For simplicity, let $\mu(e) = \mu_{k+1}(e)$, the congestion on link e after an on-line algorithm has routed all k requests. The

congestion on path P after an online algorithm has routed requests f_1, f_2, \dots, f_{j-1} is defined to be

$$\mu_j(P) = \max_{e \in P} \mu_j(e).$$

Let $\mu(P) = \mu_{k+1}(P)$. Finally, the *network congestion* after an online algorithm has routed requests f_1, f_2, \dots, f_{j-1} is defined to be

$$\mu_j = \max_{e \in E} \mu_j(e).$$

The total congestion incurred by an online algorithm is denoted $\mu = \mu_{k+1}$. Clearly, since we do not allow requests to be rejected, the congestion of a link may exceed one. If an optimal offline algorithm can service the request sequence while maintaining congestion at most one, and an online algorithm incurs congestion $c > 1$, this is an indication that the algorithm requires that the network capacity be increased by a factor of c in order to service the sequence. Alternatively, $c > 1$ may be interpreted as a degree of slowdown in service. From a theoretical point of view, the factor c is also an indication of how well the online algorithm can find suitable routes.

An online algorithm for a minimization problem is traditionally called *competitive* if it always finds a solution whose cost, or objective function value, is within a small factor of the cost incurred by an optimal offline algorithm. This technique is known as *competitive analysis* [19]. Intuitively, competitive analysis measures the degree to which the performance of an online algorithm suffers, due to its lack of knowledge about the future, compared to that of an optimal offline algorithm. Formally, consider an online algorithm A for a minimization problem, and let $\text{cost}(\sigma)$ be the cost of the solution obtained by A given request sequence σ . Let $\text{cost}^*(\sigma)$ be the cost obtained by an optimal offline algorithm given the same request sequence.

Definition 1. Online algorithm A is c competitive for a minimization problem if and only if, for all request sequences σ , $\text{cost}(\sigma) \leq c \cdot \text{cost}^*(\sigma) + a$, where a is a constant.

Definition 2. The competitive ratio of online algorithm A for a minimization problem is defined to be $\sup_{\sigma} [\text{cost}(\sigma) / \text{cost}^*(\sigma)]$.

Competitive analysis allows us to analyze online algorithms in a robust way without having to make any assumptions about the input. Since the optimal performance is fixed, this method also allows us to compare the performance of two online algorithms by using the optimal performance as a benchmark.

1.2. Related Research

There are at least three related threads of research that relate to the online routing problem we consider. We will first describe these results and then explain how our contribution is related.

Mao and Simha [23] studied three simple online list scheduling (LS) algorithms for permanent virtual circuit routing. The three online routing algorithms are described as follows:

Algorithm LS1. For request f_j , assign any route $P \in \mathcal{P}_j$.

Algorithm LS2. For request f_j , assign a route $P \in \mathcal{P}_j$ with the fewest links that have been used before.

Algorithm LS3. For request f_j , assign any route $P \in \mathcal{P}_j$ which minimizes

$$\max_{e \in E} \begin{cases} \mu_j(e), & e \notin P \\ \mu_j(e) + \frac{l_j}{c(e)}, & e \in P \end{cases}.$$

Mao and Simha showed that the competitive ratios of algorithms LS1 and LS2 are $\Omega(k)$, and the competitive ratio of algorithm LS3 is $\Omega(\sqrt{k})$ [and, implicitly, $\Omega(\sqrt{m})$, where m is the number of network links], with respect to congestion, even on two-layer directed networks with unit capacity links. Simulation experiments showed that in practice both LS2 and LS3 outperform LS1 by a large margin while LS3 constructs schedules just a little better than LS2 does. Mao and Simha left open the upper bound of LS3 and the question of whether there exist better greedy algorithms.

Aspnes et al. [1, 2] designed an elegant, asymptotically optimal online algorithm, based on techniques to approximately solve offline multicommodity flow problems [20, 21, 25], which assigns to each request a shortest path with respect to the following exponential cost function:

$$\text{cost}_e(j) = a^{\mu_j(e) + l_j/c(e)} - a^{\mu_j(e)},$$

where $a = 1 + \gamma$ for any $0 < \gamma < 1$. This algorithm, which we will call `EXP_ROUTE`, is $O(\log n)$ competitive, where $n = |V|$, on arbitrary networks.* Furthermore, Aspnes et al. proved that `EXP_ROUTE` is optimal (up to a constant factor) by showing that no algorithm can have a competitive ratio better than $(\log n - 1)/2$. An alternative proof, showing a lower bound of $(\log n + 3)/2$, was given in [17]. For switched virtual circuits, Azar et al. [9] showed how to derive a $O(\log(nT))$ competitive algorithm, where T is the

* This is technically only true if the optimal congestion is 1. For the more general case, the algorithm must be modified to work in phases, with a factor of 4 increase in the competitive ratio.

maximum duration of a virtual circuit, by concurrently using this algorithm to solve one instance of permanent virtual circuit routing for each time step. Awerbuch et al. [6] noted that the competitive ratio of any online algorithm for routing switched virtual circuits with unknown durations must be $\Omega(\sqrt[4]{n})$. However, they designed a $O(\log n)$ competitive algorithm that reroutes a request $O(\log n)$ times.

The virtual circuit routing problem has also been studied in a model in which requests may be rejected and the goal is to maximize the throughput (or, more generally, the profit) of accepted requests. When durations are unknown, no competitive algorithm exists for the problem, even on a single link and even if preemption is allowed [13]. However, Awerbuch et al. [5] designed an admission control and routing algorithm for virtual circuits with known duration. As with `EXP_ROUTE`, the algorithm first assigns to each link a cost that is an exponential function of the current congestion. Then, a request is accepted only if there exists a route whose total cost is small relative to the profit to be gained by accepting it. If the profit of a request is proportional to its throughput and the bandwidth requirement of each request is small relative to the link capacities, then the algorithm is $O(\log(nT))$ competitive on general networks. By removing the element of time, an $O(\log n)$ competitive algorithm for permanent virtual circuits can be derived from this result [24]. Without the above assumptions, randomized online algorithms with polylogarithmic competitive ratios have been designed for trees, meshes, trees of meshes, and hypercubes [7, 8, 22]. For a distributed environment, Awerbuch and Azar [3] designed randomized online algorithms with polylogarithmic competitive ratios.

The online permanent virtual circuit routing problem that we consider is a generalization of an online load balancing problem in which each in a sequence of permanent jobs must be assigned to one of n independent, parallel machines. In general, job j is specified by a tuple $(\mathbf{p}(j), a_j)$, where \mathbf{p}_j is the job's load vector and a_j is the job's arrival time. Each component $p_i(j)$ of the load vector is the weight of the job if assigned to machine i . In the *identical machines* case, the weight of every job is the same for all machines and is denoted w_j . In the *identical machines with assignment restriction* case, each $p_i(j)$ is either equal to a constant w_j or ∞ . In the *related machines* case, $p_i(j) = w_j/s(i)$. When a job arrives at the scheduler, it must be assigned immediately to exactly one machine, whose load is increased by the appropriate component of the job's load vector. The goal of any algorithm is to minimize the maximum load on the machines.

When machines are identical, the problem is the classical machine scheduling problem for which Graham [15] proved that the greedy algorithm which assigns each job to the machine with the least load is $(2 - 1/n)$ competitive. Recently, new algorithms with constant (for all n) competitive ratios strictly less than two were designed by Bartal et

al. [12] and Karger et al. [18]. For the identical machines with assignment restriction case, Azar et al. [10, 11] proved that the greedy algorithm is $\lceil \log n \rceil + 1$ competitive. They further showed that this bound is tight, up to an additive 1, by proving a lower bound of $\lceil \log(n + 1) \rceil$ for any algorithm. They also designed a randomized algorithm which is $\ln n$ competitive against an oblivious adversary and proved this result is tight as well. For the related machines case, the greedy algorithm is also $\Theta(\log n)$ competitive, but Aspnes et al. [1] showed that a nongreedy algorithm is 8 competitive. For the most general case, where the load vector is unrestricted, Aspnes et al. [1] designed a $O(\log n)$ competitive algorithm. The greedy algorithm in this case is $\Theta(n)$ competitive.

`EXP_ROUTE` is a generalization of the aforementioned $O(\log n)$ competitive online algorithm for load balancing permanent jobs in the most general unrelated machines case. Since the greedy algorithm for this load balancing problem is $\Theta(n)$ competitive, so is any greedy algorithm for the most general case of virtual circuit routing in which the load applied to a link by a request is independent of the link's capacity. However, the greedy load balancing algorithm is $\Theta(\log n)$ competitive in the identical machines with an assignment restriction case [10, 11]. This raises a natural question: Is a generalization of the greedy algorithm similarly competitive for virtual circuit generalizations of this load balancing problem? This is the question in which we are interested.

A motivation behind greedy algorithms is the possibility of reducing the control overhead involved in virtual circuit routing. `EXP_ROUTE` is capable of making excellent routing decisions but requires the computation of many exponential functions with real bases and exponents for each decision. We wonder whether algorithms that require fewer computational resources can also demonstrate sufficient efficiency in at least some nontrivial cases. In a distributed setting, the communication cost inherent in collecting network status information might dominate this improvement in local computation. However, in a centralized environment like that considered here and in essentially all the results cited previously, this communication cost is negligible since the centralized algorithm, as the only controller in the network, can assume up-to-date information at all times. In any case, an improvement in local computation cost has the potential to increase the serviceable arrival rate in a large network.

1.3. Outline

In Section 2, we will improve the lower bound given by Mao and Simha [23] for LS3 (calling it `GREEDY_ROUTE1`) and provide an upper bound with respect to congestion that is tight in many cases, including the ones considered in [23]. Specifically, we will prove that, when the bandwidth requirements of requests are approximately equal (or when various other conditions hold), the competitive ratio of

`GREEDY_ROUTE1` is $\Theta(\sqrt{\mathcal{D}m})$ on arbitrary networks, where \mathcal{D} is the maximum over all requests of the ratio of the lengths of the longest-to-shortest path for the request and m is the number of network links. We also provide a lower bound for layered networks (in which case $\mathcal{D} = 1$) which shows that the competitive ratio is $\Theta(\sqrt{m})$ in this case. When bandwidth requirements are arbitrary, the upper bounds are increased by a factor of $O(\sqrt{\mathcal{L}})$, where \mathcal{L} is the ratio of the maximum-to-minimum bandwidth requirement, although we suspect that this term does not belong in the true competitive ratio.

In Section 3, we analyze a better greedy algorithm called `GREEDY_ROUTE2` and show that it is (approximately) $O(d \log n)$ competitive, where d is the length of the longest path assigned to a request.[†] Therefore, for networks with relatively short paths relative to n , this algorithm performs well; if $d = O(\log n)$, the competitive ratio is polylogarithmic, and if $d = O(1)$, the competitive ratio is asymptotically optimal. We also prove a lower bound of $\Omega(d + \log(n - d))$ for arbitrary networks and $\Omega(d + \log(\frac{n}{d} - d))$ for layered networks. We stress that the upper bounds for both `GREEDY_ROUTE1` and `GREEDY_ROUTE2` apply to *arbitrary* networks, that is, `GREEDY_ROUTE2` is polylogarithmically competitive for *any* network with $d = O(\log n)$ rather than just those with a specific topology.

The analyses of `GREEDY_ROUTE1` and `GREEDY_ROUTE2` answer an interesting question about the generalization of the greedy algorithm to routing. On the one hand, the greedy load balancing algorithm, when applied to routing, is no longer optimal, having a competitive ratio in $\omega(\log n)$ for general networks. On the other hand, it may perform well in some cases. The primary reason that the greedy algorithms can perform poorly is that they can choose routes that are unnecessarily long, thereby increasing the congestion on too many links. Taking this into account, the factor of d in the competitive ratio of `GREEDY_ROUTE2` makes intuitive sense.

We should mention that `GREEDY_ROUTE2` is identical to the min-max algorithm simulated in [14] and compared to the min-hop algorithm and a variant of `EXP_ROUTE`. In these simulations, the goal was to maximize throughput rather than to minimize congestion. `GREEDY_ROUTE2` was shown to route up to 25% less throughput than the exponential algorithm. But the simulated network contains many routes with lengths close to n . (By inspection, it is easy to see that, for most source/destination pairs, there is a simple path that contains at least 75% of the network nodes.) We suspect that the simulation results would be different in a network with shorter paths.

[†] Our upper bound holds if the optimal route assignment has a small degree of overlap. We discuss this further in Section 3.

2. ONLINE ALGORITHM `GREEDY_ROUTE1`

In this section, we analyze the competitive ratio of `GREEDY_ROUTE1` on networks with uniform link capacity w . We formally define `GREEDY_ROUTE1` in the following way:

Algorithm `GREEDY_ROUTE1`. For request f_j , assign any route $P \in \mathcal{P}_j$ which minimizes

$$\max_{e \in E} \begin{cases} \mu_j(e), & e \notin P \\ \mu_j(e) + \frac{l_j}{w}, & e \in P \end{cases}.$$

Ties are broken arbitrarily.

In other words, `GREEDY_ROUTE1` will choose any path that does not increase the current network congestion, unless such an increase is unavoidable.

Before we can state our results formally, it will be necessary to define some notation. The following definitions hold for any particular problem instance, consisting of a request sequence $\sigma = f_1, f_2, \dots, f_k$ and a network G :

- $d_j = \min_{P \in \mathcal{P}_j} |P|$ is the length of the shortest path between nodes s_j and t_j .
- $D_j = \max_{P \in \mathcal{P}_j} |P|$ is the length of the longest path between nodes s_j and t_j .
- $\mathcal{D} = \max_{1 \leq j \leq k} (D_j/d_j)$.[‡]
- $\lambda = \min_{1 \leq j \leq k} l_j$ is the minimum bandwidth requirement among the requests in σ .
- $\Lambda = \max_{1 \leq j \leq k} l_j$ is the maximum bandwidth requirement among the requests in σ .
- $\mathcal{L} = \Lambda/\lambda$.

We will prove in Section 2.2 that the competitive ratio of `GREEDY_ROUTE1` is $O(\sqrt{\mathcal{D}\mathcal{L}m})$ on arbitrary networks, which is greater than the lower bound in [23] by a factor of $O(\sqrt{\mathcal{D}\mathcal{L}})$. In Section 2.3, we will present an improved lower bound, showing that our upper bound is, in fact, off by at most a factor of $O(\sqrt{\mathcal{L}})$, in general. The upper bound is asymptotically tight when the bandwidth requirements of the requests differ by a constant factor. We will show that the upper bound is tight when other conditions hold as well.

2.1. Trade-offs Between Time Complexity and Routing Efficiency

If we compare `GREEDY_ROUTE1` and `EXP_ROUTE` [1], we see a clear trade-off between computational requirements and routing efficiency. The routes assigned by `EXP_ROUTE` are

[‡] The value \mathcal{D} can actually be defined to be $\max_{1 \leq j \leq k} (|P_j|/d_j)$ and the results in this section will still hold. But since this quantity depends on arbitrary decisions that `GREEDY_ROUTE1` makes, it seems less desirable.

guaranteed to be asymptotically optimal, but at a relatively heavy computational price, arising from its many computations of exponential functions with real bases and exponents. On the other hand, `GREEDY_ROUTE1`, which requires only addition and comparisons, provides fast and simple decision making at the expense of the quality of the route assignment. Consider for a moment instances where \mathcal{D} and \mathcal{L} are small. For these instances, the difference between the competitive ratio of `GREEDY_ROUTE1` and that of `EXP_ROUTE` is obviously arbitrarily large if we consider arbitrarily large networks. However, if we consider real networks with less than a few hundred uniform speed links, the competitive ratios of the two algorithms do not differ significantly. When $m \leq 250$, \sqrt{m} is at most twice $\log m$, and when $m \leq 850$, \sqrt{m} is at most three times $\log m$. Even when $m = 2000$, \sqrt{m} is just slightly more than four times $\log m$.[§] The upper bound for `GREEDY_ROUTE1` also depends on the two factors $\sqrt{\mathcal{D}}$ and $\sqrt{\mathcal{L}}$. The existence of the first factor is intuitive: The algorithm does not prefer short paths to long ones (as `EXP_ROUTE` does) and therefore may add to the congestion of many more links than is necessary. For some networks though, including all layered networks, $\mathcal{D} = O(1)$, and `GREEDY_ROUTE1` is guaranteed to be $O(\sqrt{\mathcal{L}m})$ competitive. Although it seems unlikely that \mathcal{D} would be a large function of m in most networks, if this were the case, it might be reasonable to limit the paths from which `GREEDY_ROUTE1` is allowed to choose in order to minimize the error. We conjecture that the second factor does not belong in the true competitive ratio. However, if the competitive ratio of `GREEDY_ROUTE1` does depend on $\sqrt{\mathcal{L}}$, then `GREEDY_ROUTE1` is more efficient on instances consisting of requests with like bandwidth requirements. Such instances are common. For instance, in a network serving video or audio streams to customers, each request would require equal bandwidth.

2.2. An Upper Bound

In this section, we will prove our upper bound on the competitive ratio of `GREEDY_ROUTE1`:

Theorem 1. The competitive ratio of `GREEDY_ROUTE1` is $O(\sqrt{\mathcal{D}\mathcal{L}m})$.

In the proof of Theorem 1, we will use the following notation relating to an arbitrary problem instance.

[§] Since we are considering small values of m , it would be inappropriate to continue without addressing the constants hidden in the big-oh notation. For `GREEDY_ROUTE1`, the constant is only $\sqrt{2}$, and for `EXP_ROUTE`, the constant is some number strictly greater than 1, which depends on constant parameters chosen by the algorithm designer. When it is not assumed that the optimal congestion is 1, this constant increases by a factor of 4. Therefore, we can safely ignore the constants in our comparison.

- Let P_j^* denote the path assigned to request $f_j \in \sigma$ by an optimal offline algorithm.
- Let σ^* denote the optimal network congestion for the problem instance.
- Let $\tilde{\Lambda} = \Lambda/w$, where Λ is the maximum bandwidth requirement of a request and w is the capacity of the network links. Notice that

$$\mu^* \geq \tilde{\Lambda}. \quad (2)$$

- Let $\kappa = \lfloor \mu/\tilde{\Lambda} \rfloor$, where μ is the congestion of the routes assigned by GREEDY_ROUTE1. For each $i = 1, 2, \dots, \kappa$, $f_{y_i} \in \sigma$ is the first request to cause a link to have congestion at least $i\tilde{\Lambda}$ in the route assignment constructed by GREEDY_ROUTE1. In other words, for all y_i , $\mu_{y_{i+1}}(P_{y_i}) \geq i\tilde{\Lambda}$, and $\mu_h(e) < i\tilde{\Lambda}$ for all edges $e \in E$ and all $h = 1, 2, \dots, y_i$.
- For each $i = 1, 2, \dots, \kappa$, $e_{z_i} \in P_{y_i}$ is a link satisfying $\mu_{y_{i+1}}(e_{z_i}) = \mu_{y_{i+1}}(P_{y_i})$. Ties are broken in favor of the link with the smallest index.

The following three lemmas are fundamental to the proof of Theorem 1. The first two lemmas motivate our definitions of y_i and z_i . Lemma 1 states that no two indices in the set $\{y_1, y_2, \dots, y_\kappa\}$ are the same and Lemma 2 quantifies the congestion on any path $P \in \mathcal{P}_{y_i}$ just before f_{y_i} arrives and is assigned a route. Lemma 3 is a technical detail used to bound the total congestion incurred by GREEDY_ROUTE1 in terms of μ^* .

Lemma 1. For all integers h and i , $1 \leq h < i \leq \kappa$, $y_h \neq y_i$.

Proof. Assume that there exist h and i , where $1 \leq h < i \leq \kappa$, such that $y_h = y_i$. Since, by definition, $\mu_{y_{i+1}}(e_{z_i}) \geq i\tilde{\Lambda}$, we know that

$$\mu_{y_i}(e_{z_i}) \geq (i-1)\tilde{\Lambda}. \quad (3)$$

We also know, by definition, that $\mu_{y_h}(e) < h\tilde{\Lambda}$, for all $e \in E$. In particular, we know that $\mu_{y_h}(e_{z_i}) < h\tilde{\Lambda}$. But since $y_h = y_i$, $\mu_{y_i}(e_{z_i}) = \mu_{y_h}(e_{z_i}) < h\tilde{\Lambda} \leq (i-1)\tilde{\Lambda}$, which is a contradiction to (3). ■

Lemma 2. For all $i = 1, 2, \dots, \kappa$, $\mu_{y_i}(P) \geq (i-1)\tilde{\Lambda}$ for all $P \in \mathcal{P}_{y_i}$.

Proof. Suppose that there exists a path $P \in \mathcal{P}_{y_i}$ such that $\mu_{y_i}(P) < (i-1)\tilde{\Lambda}$. Since, by definition, $\mu_{y_{i+1}}(P_{y_i}) \geq i\tilde{\Lambda}$, we know that $\mu_{y_i}(P_{y_i}) \geq (i-1)\tilde{\Lambda}$. But this means that GREEDY_ROUTE1 should have assigned f_{y_i} to path P rather than P_{y_i} ($\neq P$), since this decision would have resulted in a smaller network congestion. (Notice that the network congestion did increase with request f_{y_i} by the definition of y_i .) This is a contradiction to the selection of P_{y_i} by GREEDY_ROUTE1. ■

Lemma 3. $\mathcal{D}m\mu^* \geq \sum_{e \in E} \mu(e)$.

Proof. Since no algorithm can assign a request f_j to more than D_j links, and the optimal solution had to have assigned each request to at least d_j links, it must be the case that

$$\max_{1 \leq j \leq \kappa} \frac{D_j}{d_j} \sum_{e \in E} \mu^*(e) \geq \sum_{e \in E} \mu(e).$$

The lemma follows from the fact that

$$\mathcal{D}m\mu^* \geq \max_{1 \leq j \leq \kappa} \frac{D_j}{d_j} \sum_{e \in E} \mu^*(e). \quad \blacksquare$$

We can now use the preceding lemmas to prove the main result of this section.

Proof of Theorem 1. From Lemma 2, we know that, in particular, $\mu_{y_i}(P_{y_i}^*) \geq (i-1)\tilde{\Lambda}$ for all $i = 1, 2, \dots, \kappa$. Thus,

$$\sum_{i=1}^{\kappa} \mu_{y_i}(P_{y_i}^*) \geq \sum_{i=1}^{\kappa} (i-1)\tilde{\Lambda} > \frac{\left(\frac{\mu}{\tilde{\Lambda}} - 2\right)^2 \tilde{\Lambda}}{2}. \quad (4)$$

To clarify the presentation, let us assume for the moment that the paths in the set $\{P_{y_1}^*, P_{y_2}^*, \dots, P_{y_\kappa}^*\}$ are pairwise edge disjoint. From this assumption and the fact that $y_i \leq \kappa$, we see that

$$\sum_{e \in E} \mu(e) > \sum_{e \in E} \mu_k(e) \geq \sum_{i=1}^{\kappa} \mu_{y_i}(P_{y_i}^*). \quad (5)$$

Combining this fact with (4) and Lemma 3, we see that

$$\begin{aligned} \mathcal{D}m\mu^* &\geq \sum_{e \in E} \mu(e) && \text{by Lemma 3} \\ &\geq \sum_{i=1}^{\kappa} \mu_{y_i}(P_{y_i}^*) && \text{by (5)} \\ &> \frac{\left(\frac{\mu}{\tilde{\Lambda}} - 2\right)^2 \tilde{\Lambda}}{2} && \text{by (4)}. \end{aligned}$$

Using (2), this implies that

$$\mu < \left(\sqrt{\frac{2\mathcal{D}m\mu^*}{\tilde{\Lambda}}} + 2 \right) \tilde{\Lambda} \leq (\sqrt{2\mathcal{D}m} + 2) \mu^*. \quad (6)$$

So when the paths in the set $\{P_{y_1}^*, P_{y_2}^*, \dots, P_{y_\kappa}^*\}$ are pairwise edge disjoint, GREEDY_ROUTE1 is $O(\sqrt{\mathcal{D}m})$ com-

petitive. (See Corollary 1 below.) But to prove the theorem, we must consider the more general case in which the optimal paths in the set $\{P_{y_1}^*, P_{y_2}^*, \dots, P_{y_\kappa}^*\}$ are not necessarily pairwise edge disjoint.

In general, let $I \geq 1$ be the maximum number of paths in the set $\{P_{y_1}^*, P_{y_2}^*, \dots, P_{y_\kappa}^*\}$ that intersect at an edge. Obviously,

$$\mu^* \geq \frac{I\lambda}{w}. \quad (7)$$

When $I > 1$, inequality (5) is not necessarily true because the congestion on some edges $e \in E$ may be counted up to I times in the summation on the right-hand side. However, with the aid of Lemma 1, we can modify (5) to say that

$$I \sum_{e \in E} \mu(e) > \sum_{i=1}^{\kappa} \mu_{y_i}(P_{y_i}^*). \quad (8)$$

Then, we see that

$$\begin{aligned} \mathcal{D}mI\mu^* &\geq \sum_{e \in E} \mu(e) && \text{by Lemma 3} \\ &\geq \sum_{i=1}^{\kappa} \mu_{y_i}(P_{y_i}^*) && \text{by (8)} \\ &> \frac{\left(\frac{\mu}{\bar{\Lambda}} - 2\right)^2}{2} \cdot \bar{\Lambda} && \text{by (4).} \end{aligned}$$

Using (2) and (7), this implies that

$$\mu < \left(\sqrt{\frac{2\mathcal{D}mI\mu^*}{\bar{\Lambda}}} + 2 \right) \bar{\Lambda} \leq (\sqrt{2\mathcal{D}\mathcal{L}m} + 2)\mu^*. \quad (9)$$

Thus, GREEDY_ROUTE1 is $O(\sqrt{\mathcal{D}\mathcal{L}m})$ competitive. ■

We note that there are several realistic conditions under which we can infer that the competitive ratio of GREEDY_ROUTE1 is $O(\sqrt{\mathcal{D}m})$. We define a *feasible* request sequence to be one for which there exists a set of routes with congestion $\mu^* \leq 1$. We point out that this modified definition of *competitive*, which limits consideration to a subset of instances, namely, feasible instances, has appeared previously in the literature [4].

Corollary 1. The competitive ratio of GREEDY_ROUTE1 is $O(\sqrt{\mathcal{D}m})$ if any of the following are true: (a)

- (a) $\mathcal{L} = O(1)$;
- (b) The set of optimal routes is pairwise edge disjoint;

- (c) The request sequence is feasible and $w = O(1)$ or $w = O(\lambda)$; or
- (d) $\Lambda \leq w$ and $w = O(1)$ or $w = O(\lambda)$.

Proof. Part (a) is obvious and part (b) follows from (6). For part (c), notice that when $\mu^* \leq 1$, $I\lambda \leq w$. Also, $\bar{\Lambda} \leq \mu^* \leq 1$. Substituting into (9),

$$\mu < \sqrt{2\mathcal{D}\bar{\Lambda}mI\mu^*} + 2\bar{\Lambda} \leq \sqrt{2\mathcal{D}m\frac{w}{\lambda}} + 2 = O(\sqrt{\mathcal{D}m}).$$

For part (d), notice that when $\Lambda \leq w$ it follows that $\bar{\Lambda} \leq 1$. Substituting into (9),

$$\begin{aligned} \mu &< \sqrt{2\mathcal{D}\bar{\Lambda}mI\mu^*} + 2\bar{\Lambda} \\ &\leq \sqrt{2\mathcal{D}m\frac{w}{\lambda}(\mu^*)^2} + 2 \leq O(\sqrt{\mathcal{D}m})\mu^*. \quad \blacksquare \end{aligned}$$

Notice that the first condition in part (d) must hold for real virtual circuit request sequences. Otherwise, the request with the largest bandwidth requirement cannot be assigned to any route without exceeding link capacities.

2.3. A Lower Bound

In this section, we show that the upper bound in Theorem 1 is tight up to a factor of $\sqrt{\mathcal{L}}$, in general, and tight up to a constant factor when at least one of the conditions in Corollary 1 is true. This result is formally stated as follows:

Theorem 2. The competitive ratio of GREEDY_ROUTE1 is $\Omega(\sqrt{\mathcal{D}m})$ for arbitrarily large values of \mathcal{D} and m .

To prove Theorem 2, we will construct an arbitrarily large network and corresponding request sequence and then show that if GREEDY_ROUTE1 makes bad decisions it can incur the above network congestion on that instance. On the other hand, we will show that an optimal algorithm can always find a set of pairwise edge disjoint routes for the requests, yielding unit network congestion.

2.3.1. The Network

For any integers $h > 1$ and $i \geq 1$, the network is represented by a directed graph $G_{h,i}$ containing i connected components and having a longest path with length h . All links have unit capacity. For example, $G_{3,i}$ is displayed in Fig. 1. The vertex set of $G_{h,i}$ is defined to be the union $\cup_{\iota=1}^i (U_{h,\iota} \cup V_{h,\iota})$, where $U_{h,\iota} = \{u_0^\iota, u_1^\iota, \dots, u_{h-1}^\iota\}$ contains the set of source nodes and $V_{h,\iota} = \{v_1^\iota, v_2^\iota, \dots, v_{(h-1)\iota+1}^\iota\}$ is the set of destination nodes. The arc set of $G_{h,i}$ is defined to be $\cup_{\iota=1}^i E_{h,\iota}$, where

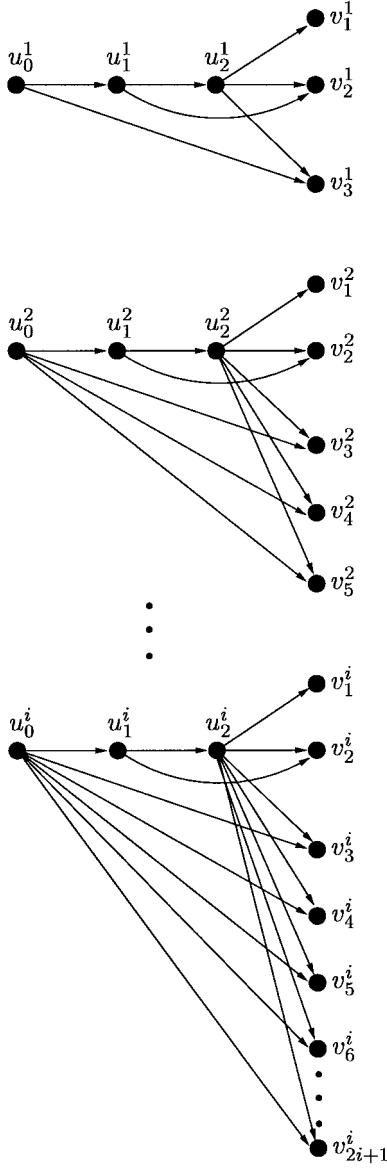


Fig. 1. The network $G_{3,i}$.

$$E_{h,\iota} = \bigcup_{\kappa=1}^{h-2} \{(u_{\kappa}^{\iota}, v_{h-\kappa}^{\iota})\} \cup \bigcup_{\kappa=1}^{(h-1)\iota+1} \{(u_{h-1}^{\iota}, v_{\kappa}^{\iota})\} \\ \cup \bigcup_{\kappa=0}^{h-2} \{(u_{\kappa}^{\iota}, u_{\kappa+1}^{\iota})\} \cup \bigcup_{\kappa=h}^{(h-1)\iota+1} \{(u_0^{\iota}, v_{\kappa}^{\iota})\}.$$

Note that

$$m = \left| \bigcup_{\iota=1}^i E_{h,\iota} \right| = (h-1)i^2 + (2h-1)i \leq (3h-2)i^2. \quad (10)$$

Notice that the i connected components of $G_{h,i}$ are the subgraphs $\{(U_{h,\iota} \cup V_{h,\iota}, E_{h,\iota}) : \iota = 1, 2, \dots, i\}$.

2.3.2. The Request Sequence

Let $\sigma_{h,i}$ denote the request sequence for the network $G_{h,i}$. Each request will arrive at time 0 and will require unit bandwidth. Therefore, we will represent each request $f_j \in \sigma_{h,i}$ simply by its (s_j, t_j) pair. The sequence $\sigma_{h,i}$ is formally defined to be the concatenation of i smaller subsequences:

$$\sigma_{h,i} = \sigma_h^1 \circ \sigma_h^2 \circ \dots \circ \sigma_h^i.$$

All paths between the source and destination of each request in subsequence σ_h^{ι} , $\iota = 1, 2, \dots, i$, are contained in the subgraph $(U_{h,\iota} \cup V_{h,\iota}, E_{h,\iota})$. Each subsequence σ_h^{ι} is further defined to be the concatenation of two subsequences A_h^{ι} and B_h^{ι} , where

$$A_h^{\iota} = (u_0^{\iota}, v_h^{\iota}), (u_0^{\iota}, v_{h+1}^{\iota}), \dots, (u_0^{\iota}, v_{(h-1)\iota+1}^{\iota})$$

and

$$B_h^{\iota} = (u_0^{\iota}, v_{h-1}^{\iota}), (u_1^{\iota}, v_{h-2}^{\iota}), \dots, (u_{h-2}^{\iota}, v_1^{\iota}).$$

As an example of this construction, consider the first three columns of Table I, which contain the requests in $\sigma_{3,3}$. (See Fig. 1 for the corresponding network $G_{3,3}$.)

2.3.3. Proof of the Lower Bound

We will use the construction in the previous two subsections to prove Theorem 2. Since the lemmas necessary for the proof are rather straightforward, we opt to state them with-

TABLE I. Request sequence $\sigma_{3,3}$ on the network $G_{3,3}$

f_j	(s_j, t_j)	P_j	P_j^*
f_1	(u_0^1, v_3^1)	$\langle u_0^1, u_1^1, u_2^1, v_3^1 \rangle$	$\langle u_0^1, v_3^1 \rangle$
f_2	(u_0^1, v_2^1)	$\langle u_0^1, u_1^1, u_2^1, v_2^1 \rangle$	$\langle u_0^1, u_1^1, v_2^1 \rangle$
f_3	(u_1^1, v_1^1)	$\langle u_1^1, u_2^1, v_1^1 \rangle$	$\langle u_1^1, u_2^1, v_1^1 \rangle$
f_4	(u_0^2, v_3^2)	$\langle u_0^2, u_1^2, u_2^2, v_3^2 \rangle$	$\langle u_0^2, v_3^2 \rangle$
f_5	(u_0^2, v_4^2)	$\langle u_0^2, u_1^2, u_2^2, v_4^2 \rangle$	$\langle u_0^2, v_4^2 \rangle$
f_6	(u_0^2, v_5^2)	$\langle u_0^2, u_1^2, u_2^2, v_5^2 \rangle$	$\langle u_0^2, v_5^2 \rangle$
f_7	(u_0^2, v_2^2)	$\langle u_0^2, u_1^2, u_2^2, v_2^2 \rangle$	$\langle u_0^2, u_1^2, v_2^2 \rangle$
f_8	(u_1^2, v_1^2)	$\langle u_1^2, u_2^2, v_1^2 \rangle$	$\langle u_1^2, u_2^2, v_1^2 \rangle$
f_9	(u_0^3, v_3^3)	$\langle u_0^3, u_1^3, u_2^3, v_3^3 \rangle$	$\langle u_0^3, v_3^3 \rangle$
f_{10}	(u_0^3, v_4^3)	$\langle u_0^3, u_1^3, u_2^3, v_4^3 \rangle$	$\langle u_0^3, v_4^3 \rangle$
f_{11}	(u_0^3, v_5^3)	$\langle u_0^3, u_1^3, u_2^3, v_5^3 \rangle$	$\langle u_0^3, v_5^3 \rangle$
f_{12}	(u_0^3, v_6^3)	$\langle u_0^3, u_1^3, u_2^3, v_6^3 \rangle$	$\langle u_0^3, v_6^3 \rangle$
f_{13}	(u_0^3, v_7^3)	$\langle u_0^3, u_1^3, u_2^3, v_7^3 \rangle$	$\langle u_0^3, v_7^3 \rangle$
f_{14}	(u_0^3, v_2^3)	$\langle u_0^3, u_1^3, u_2^3, v_2^3 \rangle$	$\langle u_0^3, u_1^3, v_2^3 \rangle$
f_{15}	(u_1^3, v_1^3)	$\langle u_1^3, u_2^3, v_1^3 \rangle$	$\langle u_1^3, u_2^3, v_1^3 \rangle$

out giving formal proofs. We first notice that an optimal algorithm can always find a pairwise edge disjoint set of routes for the requests in $\sigma_{h,i}$. Informally, this route assignment is the one in which each request in $\sigma_{h,i}$ is assigned to its shortest path in $G_{h,i}$. Formally, we have the following lemma:

Lemma 4. For all integers $h > 1$ and $i \geq 1$, the set of optimal routes for request sequence $\sigma_{h,i}$ on network $G_{h,i}$ is pairwise edge disjoint.

In the worst case, GREEDY_ROUTE1 can assign to each request its nonoptimal route (with the exception of the requests $(u_{h-2}^\iota, v_1^\iota)$, $\iota = 1, 2, \dots, i$, which each have only one route). For example, the fourth column of Table I contains the worst-case routes assigned by GREEDY_ROUTE1, while the fifth column contains an optimal route assignment. In the example, GREEDY_ROUTE1 has incurred congestion 3 after routing the requests in σ_3^1 , 5 after routing the requests in σ_3^2 , and 7 after routing the requests in σ_3^3 . In general, we have the following lemma:

Lemma 5. For all integers $h > 1$, $i \geq 1$, and $1 \leq \iota \leq i$, GREEDY_ROUTE1 can incur congestion $(h - 1)\iota + 1$ after being issued request subsequence $\sigma_h^1 \circ \sigma_h^2 \circ \dots \circ \sigma_h^\iota$ on network $G_{h,i}$.

Proof. By induction on ι . ■

We will now use Lemmas 4 and 5 to prove Theorem 2.

Proof of Theorem 2. By Lemma 5, when GREEDY_ROUTE1 is issued the request sequence $\sigma_{h,i} = \sigma_h^1 \circ \sigma_h^2 \circ \dots \circ \sigma_h^i$, it can incur congestion

$$\mu = (h - 1)i + 1. \quad (11)$$

To rewrite μ in terms of m , we first notice from (10) that

$$i \geq \sqrt{\frac{m}{3h - 2}}. \quad (12)$$

Substituting (12) into (11), we see that

$$\mu \geq (h - 1) \left(\sqrt{\frac{m}{3h - 2}} \right) + 1 = \Omega(\sqrt{\mathcal{D}m}).$$

The last equality follows from the fact that, for every request $f_j \in A_h^\iota$, $\iota = 1, 2, \dots, i$, $d_j = 1$, $D_j = h$, and these are the minimum and maximum such values possible in the network $G_{h,i}$. Thus, since $\mu^* = 1$ by Lemma 4, the competitive ratio of GREEDY_ROUTE1 is $\Omega(\sqrt{\mathcal{D}m})$. ■

According to Theorem 1, GREEDY_ROUTE1 is $O(\sqrt{\mathcal{L}m})$ competitive on layered networks since the length of every path for any request is the same, causing $\mathcal{D} = 1$. We can show that Theorem 1 is tight up to a factor of $O(\sqrt{\mathcal{L}})$ on L -layered networks, even on an *end-to-end* instance in which every path has length L . Specifically, we can prove the following theorem, whose proof we omit.

Theorem 3. The competitive ratio of GREEDY_ROUTE1 is $\Omega(\sqrt{m})$ on layered networks for arbitrarily large m .

This result also answers a relevant question about the way GREEDY_ROUTE1 breaks ties. GREEDY_ROUTE1, as defined, breaks ties arbitrarily: It may choose any route that satisfies the requirements of the algorithm. Thus, the upper bound of Theorem 1 holds for any tie-breaking scheme. In practice, however, it would be reasonable to choose the shortest route that satisfies the algorithm. In this way, the fewest links are affected by each decision and requests will presumably reach their destinations sooner. However, this result shows that if GREEDY_ROUTE1 uses this tie-breaking scheme, its competitive ratio is still $\Omega(\sqrt{m})$. Notice that this result also holds for an algorithm which first narrows its consideration to the set of shortest routes and then uses GREEDY_ROUTE1 to choose from among those routes.

3. ONLINE ALGORITHM GREEDY_ROUTE2

In this section, we consider a more refined greedy online algorithm for routing permanent virtual circuits.[#] GREEDY_ROUTE2 differs from GREEDY_ROUTE1 in that it bases its routing decision only on the maximum congestion of the routes that can be assigned to the request, rather than on the maximum congestion in the network. We will show that there are classes of networks on which GREEDY_ROUTE2 is guaranteed to have a competitive ratio that is polylogarithmic in the number of network nodes. Furthermore, GREEDY_ROUTE2 is as simple and fast as GREEDY_ROUTE1 in the worst case. The online algorithm that we study in this section is formally defined as follows:

Algorithm GREEDY_ROUTE2. For request f_j , assign any route $P \in \mathcal{P}_j$ which minimizes $\max_{e \in P} \left\{ \mu_j(e) + \frac{l_j}{w} \right\}$. Ties are broken arbitrarily.

In other words, GREEDY_ROUTE2 considers all routes for a request and assigns the one that would have the minimum congestion if the request were assigned to it.

An adversary like the one used to prove Theorem 2

[#] A preliminary version of some of the results in this section appeared in [16].

cannot fool GREEDY_ROUTE2 into choosing bad routes in the same way that it can fool GREEDY_ROUTE1. Consider what happens when GREEDY_ROUTE2 is presented with the constructions used to prove Theorems 2 and 3. When presented with network $G_{h,1}$ and request sequence $\sigma_{h,1}$ (in the proof of either Theorem 2 or 3), GREEDY_ROUTE2 can incur congestion h in the same way that GREEDY_ROUTE1 can. But when presented with network $G_{h,i}$ and request sequence $\sigma_{h,i}$, for $i > 1$, GREEDY_ROUTE2 will incur congestion at most h , whereas GREEDY_ROUTE1 could incur congestion $(h - 1)i + 1$. This proves only that the competitive ratio of GREEDY_ROUTE2 is $\Omega(m)$, where m is the number of network links, if the length of the longest path is $\Omega(m)$, in effect, proving a lower bound on the order of the longest path length $d = \max_j \max_{P \in \mathcal{P}_j} |P|$. In Section 3.3, we will strengthen this result by showing that the competitive ratio of GREEDY_ROUTE2 is $\Omega(d + \log(n - d))$. Therefore, the greedy algorithm, when applied to routing, no longer has a logarithmic competitive ratio for general instances.

We can also use a result from the previous section to infer an upper bound on the competitive ratio of GREEDY_ROUTE2, since Theorem 1 holds for GREEDY_ROUTE2 just as it does for GREEDY_ROUTE1. The proof of this fact is identical to the proof of Theorem 1, primarily because Lemma 2 holds for GREEDY_ROUTE2 also. We restate this result for GREEDY_ROUTE2 here for completeness.

Theorem 4. The competitive ratio of GREEDY_ROUTE2 is $O(\sqrt{\mathcal{D}\mathcal{L}m})$.

In Section 3.2, we will show that a stronger upper bound on the competitive ratio of GREEDY_ROUTE2 is possible on some networks. Specifically, we will prove that the competitive ratio of GREEDY_ROUTE2 is $O(d \log n)$ if there exist pairwise edge disjoint routes for the requests. (The actual optimal path requirement is slightly less strict and arises from the proof technique. The same result holds if the ratio of the maximum to minimum bandwidth requirement is constant and the maximum number of optimal paths that intersect is bounded by a polynomial in n .) We believe that a similar (or better) upper bound will still hold if the optimal paths are not disjoint. We provide some intuitive support for this conjecture at the end of Section 3.2.

Consider for a moment instances that obey this optimal route requirement. In this context, our result indicates that if the network has paths whose lengths are $O(\log n)$ (e.g., splitter networks) then the competitive ratio of GREEDY_ROUTE2 is guaranteed to be a polylogarithmic function of the number of network nodes, which comes very close to matching the problem lower bound of $\Omega(\log n)$. If the length of the paths is constant, then GREEDY_ROUTE2 is asymptotically optimal.

3.1. Trade-offs Between Time Complexity and Routing Efficiency

Similar to GREEDY_ROUTE1, GREEDY_ROUTE2 requires only comparisons and additions to assign a route to request f_j . However, the best-case time for GREEDY_ROUTE2 is the same as the worst case, unlike GREEDY_ROUTE1 which does not necessarily have to consider every route. As was discussed in the previous section, both algorithms make their decisions more quickly than EXP_ROUTE [1]. The real-time difference would be accentuated on networks in which a large number of exponential computations is required.

As was the case with GREEDY_ROUTE1, GREEDY_ROUTE2 can be expected to perform worse than EXP_ROUTE when paths can be relatively long. This factor of d makes intuitive sense in the competitive ratio of GREEDY_ROUTE2 when one compares GREEDY_ROUTE2 to EXP_ROUTE. When all routes for a request have unit length, both algorithms will choose a route with the minimum congestion. As the maximum lengths of the routes increase, EXP_ROUTE will be more discriminating than GREEDY_ROUTE2 and the gap between their competitive ratios will increase. This can be illustrated with a few examples. First, suppose that both algorithms are confronted with a choice of two routes—one containing one link with congestion c and another containing 100 links with maximum congestion c . Whereas GREEDY_ROUTE2 will not discriminate between the two routes, EXP_ROUTE will choose the shortest one. EXP_ROUTE will be more discriminating even if the two paths were to have the same length. For example, consider two paths with five links. The first path has congestion 5 on one link and no congestion on any other link. The second path has congestion 5, 4, 3, 2, and 1 on its five links. EXP_ROUTE will clearly choose the first path, which is intuitively a much better choice, while GREEDY_ROUTE2 will not discriminate. We quantify in this section the degree to which the competitive ratios of GREEDY_ROUTE2 and EXP_ROUTE differ.

It seems reasonable to expect large networks with arbitrary topologies to have relatively short paths between most nodes. For example, by using the traceroute program, one can see that between most pairs of randomly selected IP addresses in the United States there is a site-to-site route containing at most 15 nodes. While one could probably concoct a route with many more nodes, only these relatively short routes are used in practice. By restricting GREEDY_ROUTE2 or GREEDY_ROUTE1 to a set of relatively short routes *a priori*, one would expect good performance based on our results for these algorithms (assuming the optimal algorithm is also restricted to these routes). This method of considering restricted sets of paths might be used with success on any network with small diameter. Another good scenario for GREEDY_ROUTE2 might be one in which the traffic pattern favors source/destination pairs which have short routes.

3.2. An Upper Bound

In this section, we prove our new upper bound on the competitive ratio of `GREEDY_ROUTE2`. The method that we use to prove the result is adapted from a technique used by Azar et al. [10, 11] to show that the greedy algorithm is $O(\log n)$ competitive for an online 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 online 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 showed 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 online congestion caused by `GREEDY_ROUTE2` on each *optimal path* and partition the congestion on each of these optimal paths into layers with width at most $\tilde{\Lambda}$, the maximum bandwidth requirement divided by the capacity of the network links. 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.

3.2.1. Notation

Before formally stating our theorem and its proof, we will present the notation that we will use, along with several useful facts:

- $d = \max_{1 \leq j \leq k} D_j$ is the length of the longest path that can be assigned to any request.
- $\tilde{\Lambda} = \Lambda/w$, where w is the width of the network links. Also, for any j , $1 \leq j \leq k$, let $\tilde{l}_j = l_j/w$.

Fact 1. $\mu^* \geq \tilde{\Lambda}$.

Fact 2. $\mu^* \geq 1/m \sum_{j=1}^k (d_j \tilde{l}_j)$.

- $e_j^* \in P_j^*$ is an edge $e \in P_j^*$ satisfying $\mu(e) = \mu(P_j^*)$. Ties are broken in favor of the link with the smallest index.
- $\mathcal{F} = \max_{e \in E} |\{j : e_j^* = e, 1 \leq j \leq k\}|$ is the maximum number of optimal paths P_j^* whose edge e_j^* is the same.

The following fact is true since \mathcal{F} is a lower bound on the maximum number of optimal paths that intersect at the same edge.

Fact 3. $\mu^* \geq (\mathcal{F}\lambda)/w$.

- For all $i \geq 1$, $1 \leq j \leq k$,

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

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

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

- W_{ij}^r is the portion of W_{ij} on edge e_j^* that was added by request f_r , $1 \leq r \leq k$, in the online route assignment (divided by w).

The following fact is true because if $W_{ij}^r > 0$ for more than two layers then $\tilde{l}_r > \tilde{\Lambda}$, 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 online route assignment.

The following fact is true because f_j traverses $|P_j| \leq D_j$ links, each of which may be e_r^* for at most \mathcal{F} optimal paths P_r^* .

Fact 6. For all i and j , $|S_{ij}| \leq |S_{0j}| \leq D_j \mathcal{F}$.

- $R_{ij} = \sum_{r \in S_{ij}} (\tilde{l}_j - W_{ir}^j)$ is the total congestion incurred by f_j in all layers greater than i .

Let us explain this definition in more detail. The quantity that we are describing is at most equal to \tilde{l}_j times the number of edges e_r^* to which f_j adds congestion after layer i (which is $|S_{ij}| \cdot \tilde{l}_j$). We must say "at most" because it is possible, if f_j adds congestion to layer $i+1$ on some edge e_r^* , that some congestion was also added to layer i . (No congestion could have been added to a layer before i by Fact 5.) Thus, to get R_{ij} , we must subtract off this quantity, W_{ir}^j , for each edge on which this is the case. On the other hand, if f_j does not add any congestion to any edges e_r^* in layer $i+1$, then $R_{ij} = |S_{ij}| \cdot \tilde{l}_j$. The equation above takes into account both of these possibilities because in the second case $W_{ir}^j = 0$.

Fact 7. For all i and j , $R_{ij} \leq |S_{ij}| \cdot \tilde{l}_j \leq D_j \mathcal{F} \tilde{\Lambda}$.

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

The next fact follows simply from the definitions. On the other hand, Fact 9 requires an explicit proof.

Fact 8. $R_0 \leq \mathcal{F} \sum_{j=1}^k (D_j \tilde{l}_j)$.

Fact 9. $R_i = \sum_{l>i} W_l$.

Proof. Notice that

$$\begin{aligned}
 R_i &= \sum_{j=1}^k R_{ij} && \text{by the definition of } R_i \\
 &= \sum_{j=1}^k \sum_{r \in S_{ij}} (\tilde{l}_j - W_{ir}^j) && \text{by the definition of } R_{ij} \\
 &= \sum_{r=1}^k \sum_{j: r \in S_{ij}} (\tilde{l}_j - W_{ir}^j) && \text{by changing the order of summation} \\
 &= \sum_{r=1}^k \sum_{l>i} W_{lr} && \text{(see below)} \\
 &= \sum_{l>i} \sum_{r=1}^k W_{lr} && \text{by changing the order of summation} \\
 &= \sum_{l>i} W_l && \text{by the definition of } W_l.
 \end{aligned}$$

A more lengthy explanation is in order for the fourth equality. The term $\sum_{j: r \in S_{ij}} (\tilde{l}_j - W_{ir}^j)$ refers to the total bandwidth of all requests f_j (divided by w) that cross e_r^* after layer i , minus the portions that fall in layer i . In other words, this is the total congestion on edge e_r^* after layer i . (See the text accompanying the definition of R_{ij} for an explanation of the “ $-W_{ir}^j$ ”.) By definition, this quantity is simply the total width of all the layers of e_r^* after layer i , or $\sum_{l>i} W_{lr}$. ■

The last fact follows from Fact 9:

Fact 10. $R_i = R_{i-1} - W_i$.

3.2.2. Proof of the Upper Bound

We can now formally state our upper-bound result for `GREEDY_ROUTE2`:

Theorem 5. The competitive ratio of `GREEDY_ROUTE2` is $\left(d \min\{\mathcal{L}, \mathcal{F}\} \log\left(\frac{n\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}}\right) \right)$.

We will use the following three lemmas to prove Theorem 5:

Lemma 6. $\mu \leq \max_j \mu(P_j^*) + \tilde{\Lambda}$.

Proof. Consider an edge $e \in E$ which satisfies $\mu(e) = \mu$. Let f_y be the last request assigned to a path containing e (so $e \in P_y$). Notice that since $\tilde{\Lambda} \geq \tilde{l}_y$

$$\mu_y(P_y) = \mu_y(e) = \mu - \tilde{l}_y \geq \mu - \tilde{\Lambda}. \quad (13)$$

Also, by the definition of `GREEDY_ROUTE2`, we know that, for all $P \in \mathcal{P}_y$, including P_y^* ,

$$\mu_y(P) \geq \mu_y(P_y^*). \quad (14)$$

Combining (13) and (14), we see that $\mu_y(P_y^*) \geq \mu - \tilde{\Lambda}$, which implies that

$$\mu \leq \mu_y(P_y^*) + \tilde{\Lambda} \leq \max_j \mu(P_j^*) + \tilde{\Lambda}. \quad \blacksquare$$

Lemma 7. For all i and j , $D_j \mathcal{F} \cdot W_{ij} \geq R_{ij}$.

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 left-hand side is always nonnegative.

CASE 2. $W_{ij} = \tilde{\Lambda}$.

In this case, the lemma is also clearly true since, by Fact 7, $R_{ij} \leq D_j \mathcal{F} \tilde{\Lambda}$.

CASE 3. $R_{ij} > 0$ and $W_{ij} < \tilde{\Lambda}$.

Since $R_{ij} > 0$, we know, by definition, that $S_{ij} \neq \emptyset$. 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 $W_{ij} < \tilde{\Lambda}$ the i th layer of e_r^* is the last nonzero layer, and, thus, $\mu(P_j^*) < i\tilde{\Lambda}$ by the definition of W_{ij} . So, clearly, $\mu_j(P_j^*) < i\tilde{\Lambda}$, and since $\mu_j(P_j) \leq \mu_j(P_j^*)$ by the definition of `GREEDY_ROUTE2` and $e_r^* \in P_j$, we know that

$$\mu_j(e_r^*) < i\tilde{\Lambda}. \quad (15)$$

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

$$\mu_{j+1}(e_r^*) > i\tilde{\Lambda}. \quad (16)$$

In other words, (15) and (16) tell us that just before f_j was assigned to route P_j the edge $e_r^* \in P_j$ had congestion less than $i\tilde{\Lambda}$ [but greater than $(i-1)\tilde{\Lambda}$ since $\tilde{l}_j \leq \tilde{\Lambda}$], and just after f_j was assigned, e_r^* had congestion greater than $i\tilde{\Lambda}$. Thus, the value \tilde{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,

$$\mu_j(P_j) \geq \mu_j(e_r^*) = (i - 1)\tilde{\Lambda} + (\tilde{\Lambda} - W_{ir}^j). \quad (17)$$

We next need to show that

$$W_{(i-1)j} = \tilde{\Lambda}. \quad (18)$$

Assume, for contradiction, that $W_{(i-1)j} < \tilde{\Lambda}$. This implies that $\mu(P_j^*) < (i - 1)\tilde{\Lambda}$, and, therefore, $\mu_j(P_j^*) < (i - 1)\tilde{\Lambda}$. But this means that f_j should not have been assigned to P_j since $\mu_j(P_j) > (i - 1)\tilde{\Lambda}$. Thus, (18) is true. Now, by combining (18) with the assumption that $W_{ij} < \tilde{\Lambda}$, we see that

$$\mu(P_j^*) = (i - 1)\tilde{\Lambda} + W_{ij}. \quad (19)$$

Using the facts above, we can deduce the following:

$$\begin{aligned} (i - 1)\tilde{\Lambda} + W_{ij} &= \mu(P_j^*) && \text{by (19)} \\ &\geq \mu_j(P_j^*) \\ &\geq \mu_j(P_j) && \text{by the definition of} \\ &\geq (i - 1)\tilde{\Lambda} + (\tilde{\Lambda} - W_{ir}^j) && \text{by (17).} \end{aligned}$$

Thus,

$$W_{ij} \geq \tilde{\Lambda} - W_{ir}^j. \quad (20)$$

Finally, we can conclude that

$$\begin{aligned} R_{ij} &= \sum_{r \in S_{ij}} (\tilde{I}_j - W_{ir}^j) && \text{by the definition of } R_{ij} \\ &\leq \sum_{r \in S_{ij}} (\tilde{\Lambda} - \min_{r' \in S_{ij}} W_{ir'}^j) \\ &= |S_{ij}| (\tilde{\Lambda} - \min_{r' \in S_{ij}} W_{ir'}^j) \\ &\leq D_j \mathcal{F} \cdot W_{ij} && \text{by Fact 6 and (20).} \end{aligned} \quad \blacksquare$$

Lemma 8. For any i ,

$$R_i \leq \left(\frac{d\mathcal{F}}{d\mathcal{F} + 1} \right)^i R_0.$$

Proof. By Lemma 7 and the definitions of W_i and R_i , we know that

$$W_i = \sum_{j=1}^k W_{ij} \geq \frac{1}{\mathcal{F}} \sum_{j=1}^k \frac{R_{ij}}{D_j} \geq \frac{1}{d\mathcal{F}} \sum_{j=1}^k R_{ij} = \frac{1}{d\mathcal{F}} R_i. \quad (21)$$

From (21) and Fact 10, it follows that $R_i \leq R_{i-1} - [1/(d\mathcal{F})]R_i$. Therefore,

$$R_i \leq \frac{d\mathcal{F}}{d\mathcal{F} + 1} R_{i-1}. \quad (22)$$

Finally, by recursively applying inequality (22), we conclude that

$$R_i \leq \left(\frac{d\mathcal{F}}{d\mathcal{F} + 1} \right)^i R_0. \quad \blacksquare$$

Finally, we can prove the main result of this section:

Proof of Theorem 5. First, let

$$b = \left\lceil \log_{(d\mathcal{F}+1)/d\mathcal{F}} \left(\frac{m\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}} \right) \right\rceil \geq \log_{(d\mathcal{F}+1)/d\mathcal{F}} \left(\frac{m\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}} \right).$$

Thus,

$$\left(\frac{d\mathcal{F} + 1}{d\mathcal{F}} \right)^b \geq \frac{m\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}}. \quad (23)$$

Then, we can deduce that

$$\begin{aligned} R_b &\leq \left(\frac{d\mathcal{F}}{d\mathcal{F} + 1} \right)^b R_0 && \text{by Lemma 8} \\ &\leq \frac{1}{\left(\frac{d\mathcal{F} + 1}{d\mathcal{F}} \right)^b} \mathcal{F} \sum_{j=1}^k (D_j \tilde{I}_j) && \text{by Fact 8} \\ &\leq \frac{\min\{\mathcal{L}, \mathcal{F}\}}{m\mathcal{F}} \mathcal{F} \sum_{j=1}^k (D_j \tilde{I}_j) && \text{by (23)} \\ &\leq \mathcal{D} \min\{\mathcal{L}, \mathcal{F}\} \frac{\sum_{j=1}^k (d_j \tilde{I}_j)}{m} \\ &\leq \mathcal{D} \min\{\mathcal{L}, \mathcal{F}\} \mu^* && \text{by Fact 2.} \end{aligned} \quad (24)$$

Now notice that, for all j ,

$$\begin{aligned} \mu(P_j^*) &= \sum_i W_{ij} && \text{by Fact 4} \\ &= \sum_{i=1}^b W_{ij} + \sum_{i>b} W_{ij} \\ &\leq \sum_{i=1}^b W_{ij} + \sum_{i>b} W_i \\ &= \sum_{i=1}^b W_{ij} + R_b && \text{by Fact 9} \\ &\leq b\tilde{\Lambda} + \mathcal{D} \min\{\mathcal{L}, \mathcal{F}\} \mu^* && \text{by (24) and the} \\ &&& \text{definition of } W_{ij}. \end{aligned}$$

Thus,

$$\begin{aligned} \max_j \mu(P_j^*) &\leq b\tilde{\Lambda} + \mathcal{D} \min\{\mathcal{L}, \mathcal{F}\} \mu^* \\ &= \left\lceil \log_{(d\mathcal{F}+1)/d\mathcal{F}} \left(\frac{m\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}} \right) \right\rceil \tilde{\Lambda} + \mathcal{D} \min\{\mathcal{L}, \mathcal{F}\} \mu^*. \end{aligned}$$

By Lemma 6, this implies that

$$\mu \leq \left\lceil \log_{(d\mathcal{F}+1)/d\mathcal{F}} \left(\frac{m\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}} \right) \right\rceil \tilde{\Lambda} + \tilde{\Lambda} + \mathcal{D} \min\{\mathcal{L}, \mathcal{F}\} \mu^*.$$

So, by Facts 1 and 3 and Lemma 12 (in the Appendix), the competitive ratio of GREEDY_ROUTE2 is

$$\begin{aligned} \frac{\mu}{\mu^*} &\leq \frac{\left\lceil \log_{(d\mathcal{F}+1)/d\mathcal{F}} \left(\frac{m\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}} \right) \right\rceil \tilde{\Lambda} + \tilde{\Lambda}}{\mu^*} + \mathcal{D} \min\{\mathcal{L}, \mathcal{F}\} \\ &\leq \frac{\left\lceil \log_{(d\mathcal{F}+1)/d\mathcal{F}} \left(\frac{m\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}} \right) \right\rceil \tilde{\Lambda} + \tilde{\Lambda}}{\max\left\{ \frac{\mathcal{F}\lambda}{w}, \tilde{\Lambda} \right\}} + \mathcal{D} \min\{\mathcal{L}, \mathcal{F}\} \\ &\leq \frac{d\mathcal{F}\tilde{\Lambda} \log\left(\frac{m\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}} \right) + 2\tilde{\Lambda}}{\max\left\{ \frac{\mathcal{F}\lambda}{w}, \tilde{\Lambda} \right\}} + \mathcal{D} \min\{\mathcal{L}, \mathcal{F}\} \\ &= O\left(d \min\{\mathcal{L}, \mathcal{F}\} \log\left(\frac{n\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}} \right) \right). \quad \blacksquare \end{aligned}$$

Combining Theorems 4 and 5, we have the following result:

Theorem 6. The competitive ratio of GREEDY_ROUTE2 is

$$O\left(\min\left\{ d \min\{\mathcal{L}, \mathcal{F}\} \log\left(\frac{n\mathcal{F}}{\min\{\mathcal{L}, \mathcal{F}\}} \right), \sqrt{\mathcal{D}\mathcal{L}m} \right\} \right).$$

We can state the following corollaries giving tighter upper bounds for circumstances in which the optimal congestion is small. The first result follows directly from Theorem 5.

Corollary 2. On instances in which the optimal routes are pairwise edge disjoint, the competitive ratio of GREEDY_ROUTE2 is $O(d \log n)$.

The second corollary is more general. Recall that a

feasible request sequence is one for which there exists a route assignment with congestion $\mu^* \leq 1$.

Corollary 3. On feasible request sequences, the competitive ratio of GREEDY_ROUTE2 is

$$\begin{cases} O\left(d\mathcal{L} \log\left(\frac{nw}{\Lambda} \right) \right) & \mathcal{L} \leq \mathcal{F} \\ O\left(d \frac{w}{\lambda} \log n \right), & \mathcal{F} \leq \mathcal{L} \end{cases} = O\left(d \frac{w}{\lambda} \log\left(\frac{nw}{\Lambda} \right) \right).$$

Proof. If $\mu^* \leq 1$, then by Facts 1 and 3, $\Lambda \leq w$ and $\mathcal{F} \leq w/\lambda$. The corollary follows by substituting into the result in Theorem 5. \blacksquare

On feasible instances in which the minimum required bandwidth of a request is at least a constant fraction of the link capacity w , the competitive ratio of GREEDY_ROUTE2 is $O(d \log n)$. Specifically, suppose that, for all j , $l_j \geq w/\beta$, for some $\beta = O(1)$. Then, $(w/\Lambda) \geq (w/\lambda) \geq \beta$, and the competitive ratio of GREEDY_ROUTE2 is $O(d\beta \log(\beta n)) = O(d \log n)$. We can also claim that the competitive ratio of GREEDY_ROUTE2 is $O(d \log n)$ on feasible instances if $w = O(1)$.

3.2.3. A Final Note

The existence of the factor \mathcal{F} in the previous theorem is not intuitive and we do not think that it belongs in the competitive ratio of GREEDY_ROUTE2. It is easy to show that when the problem instance is such that all of the optimal paths intersect (and $\mathcal{F} = k$) the competitive ratio of GREEDY_ROUTE2 is at most \mathcal{L} . To see this, notice that when all of the optimal paths intersect, the optimal congestion is at least $(k\lambda)/w$. The congestion of any route assignment is at most $(k\Lambda)/w$. Thus, the competitive ratio of any algorithm is at most \mathcal{L} .

Theorem 7. If the optimal paths for an instance all intersect at one link, then any algorithm is \mathcal{L} competitive for that instance.

This result gives credence to the idea that as \mathcal{F} increases the competitive ratio decreases, contrary to the theorem.

3.3. A Lower Bound

In this section, we present a lower bound on the competitive ratio of GREEDY_ROUTE2. Our lower bound improves upon the lower bound of d that can be inferred from the construction in the proof of Theorem 2. For small values of d , the lower bound is close to the upper bound in Theorem 5. Our experience indicates that the true competitive ratio of GREEDY_ROUTE2 may be closer to this lower bound than to the upper bound in Theorem 5.

Theorem 8. The competitive ratio of GREEDY_ROUTE2 is $\Omega(d + \log(n - d))$ for arbitrarily large values of d and n .

To prove the theorem, we will first construct an arbitrarily large network $G_{h,i}$ (for any $h \geq 1$ and $i \geq 1$) and a corresponding request sequence that can cause GREEDY_ROUTE2 to incur congestion equal to $h + i$. We will then show that this quantity is equivalent to the quantity in Theorem 8.

3.3.1. The Network

For any integers $h \geq 1$ and $i \geq 1$, $G_{h,i}$ is a directed network with unit capacity links. The vertex set of $G_{h,i}$ is defined to be

$$V(G_{h,i}) = \{u_\iota : 1 \leq \iota \leq 2^i - 1\} \cup \{x_{0,\iota} : 1 \leq \iota \leq 2^i\} \\ \cup \{x_{\iota,2^i} : 1 \leq \iota \leq h - 1\} \cup \{v_\iota : 1 \leq \iota \leq h\}.$$

Notice that $G_{h,i}$ has

$$n = 2^{i+1} + 2h - 2 \quad (25)$$

vertices. The directed edge set of $G_{h,i}$ is defined to be

$$E(G_{h,i}) = \bigcup_{\iota=1}^i \{(u_{2^{\iota-1}}, x_{0,2^{\iota-1}}), (u_{3 \cdot 2^{\iota-1}}, x_{0,3 \cdot 2^{\iota-1}}), \\ (u_{5 \cdot 2^{\iota-1}}, x_{0,5 \cdot 2^{\iota-1}}), \dots, (u_{(2^i - 2^{\iota-1})}, x_{0,(2^i - 2^{\iota-1})})\} \\ \cup \bigcup_{\iota=1}^i \{(u_{2^{\iota-1}}, x_{0,2 \cdot 2^{\iota-1}}), (u_{3 \cdot 2^{\iota-1}}, x_{0,4 \cdot 2^{\iota-1}}), \\ (u_{5 \cdot 2^{\iota-1}}, x_{0,6 \cdot 2^{\iota-1}}), \dots, (u_{(2^i - 2^{\iota-1})}, x_{0,2^i})\} \\ \cup \{(x_{\iota,2^i}, x_{\iota+1,2^i}) : 0 \leq \iota \leq h - 2\} \\ \cup \{(x_{0,\iota}, v_1) : 1 \leq \iota \leq 2^i - 1\} \cup \{(x_{h-1,2^i}, v_1)\} \\ \cup \{(x_{\iota,2^i}, v_{\iota+1}) : 1 \leq \iota \leq h - 1\} \\ \cup \{(v_1, v_\iota) : 2 \leq \iota \leq h\}.$$

As an example, $G_{4,3}$ is shown in Fig. 2.

3.3.2. The Request Sequence

We will define the request sequence $\sigma_{h,i}$ for network $G_{h,i}$ to be the concatenation of two subsequences σ_i and σ_h^i . Each request f_j requires unit bandwidth and has arrival time 0, and so we denote each request simply by (s_j, t_j) . The subsequence σ_i consists of $2^i - 1$ requests organized into i phases. In phase ι , $1 \leq \iota \leq i$, we issue the $2^{i-\iota}$ requests $(u_{2^{\iota-1}}, v_1)$, $(u_{3 \cdot 2^{\iota-1}}, v_1)$, $(u_{5 \cdot 2^{\iota-1}}, v_1)$, \dots , $(u_{(2^i - 2^{\iota-1})}, v_1)$. The request subsequence σ_h^i contains the following h requests: $(x_{0,2^i}, v_2)$, $(x_{1,2^i}, v_3)$, \dots , $(x_{h-2,2^i}, v_h)$, $(x_{h-1,2^i}, v_h)$.

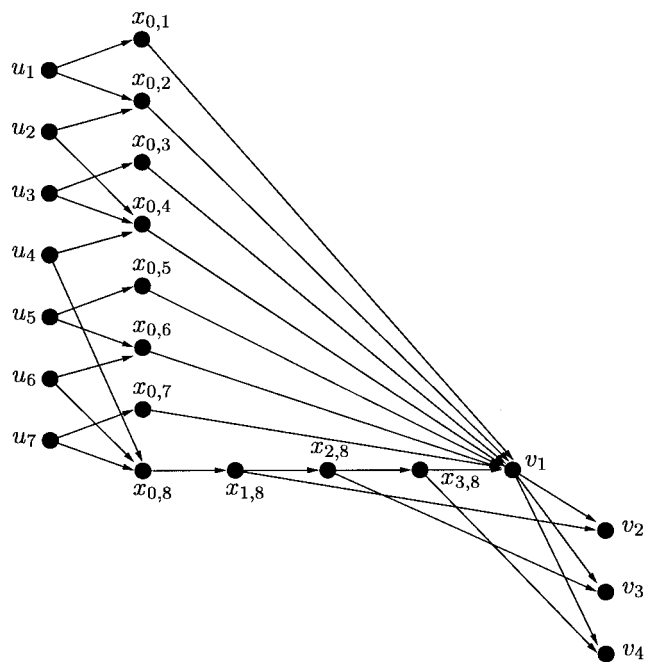


Fig. 2. The network $G_{4,3}$.

v_1). Notice that the longest path that can be assigned to a request in $G_{h,i}$ has length

$$d = h + 1. \quad (26)$$

3.3.3. Proof of the Lower Bound

We can now use the preceding construction to prove Theorem 8. To this end, we first present three lemmas that describe the behavior of GREEDY_ROUTE2 and the optimal algorithm when they are given this instance. Lemma 9 and Lemma 10 together imply that GREEDY_ROUTE2 can incur congestion $h + i$. Lemma 11 states that the optimal route assignment has congestion 1. As in Subsection 2.3.3, the proofs for these lemmas are omitted.

Lemma 9. Suppose that we issue the request sequence σ_i on network $G_{h,i}$, for any integers $h \geq 1$ and $i \geq 1$. At the end of phase ι , $1 \leq \iota \leq i$, GREEDY_ROUTE2 can incur congestion ι on each of the links in

$$\{(x_{0,2^i}, v_1), (x_{0,2 \cdot 2^i}, v_1), (x_{0,3 \cdot 2^i}, v_1), \dots, (x_{0,2^i - 2^i}, v_1)\} \\ \cup \{(x_{0,2^i}, x_{1,2^i}), (x_{1,2^i}, x_{2,2^i}), \dots, (x_{h-2,2^i}, x_{h-1,2^i}), (x_{h-1,2^i}, v_1)\}$$

and no link in the network can have congestion greater than ι .

Proof. By induction on ι . ■

Lemma 10. For any integer $h \geq 1$ and $i \geq 1$, suppose that every link of network $G_{h,i}$ in the set

$$\{(x_{0,2^i}, x_{1,2^i}), (x_{1,2^i}, x_{2,2^i}), \dots, (x_{h-2,2^i}, x_{h-1,2^i}), (x_{h-1,2^i}, v_1)\}$$

has congestion c and every other link in the network has congestion at most c . Then, if we issue the requests in request sequence σ_h^i , GREEDY_ROUTE2 can assign routes such that, after the j th request in σ_h^i , every link in the set

$$\{(x_{j-1,2^i}, x_{j,2^i}), (x_{j,2^i}, x_{j+1,2^i}), \dots, (x_{h-2,2^i}, x_{h-1,2^i}), (x_{h-1,2^i}, v_1)\}$$

will have congestion $c + j$ and no link in the network will have congestion greater than $c + j$.

Proof. By induction on j . ■

Lemma 11. For all integers $h > 1$ and $i \geq 1$, the set of optimal routes for request sequence $\sigma_{h,i}$ on network $G_{h,i}$ is pairwise edge disjoint.

Finally, we can use the preceding three lemmas to prove Theorem 8.

Proof of Theorem 8. Recall that the request sequence $\sigma_{h,i}$ consists of the concatenation of the subsequences σ_i and σ_h^i . By Lemma 9, GREEDY_ROUTE2 can incur congestion i after the i th (and last) phase of subsequence σ_i . Specifically, GREEDY_ROUTE2 can incur congestion i on every link in the set

$$\{(x_{0,2^i}, x_{1,2^i}), (x_{1,2^i}, x_{2,2^i}), \dots, (x_{h-2,2^i}, x_{h-1,2^i}), (x_{h-1,2^i}, v_1)\}.$$

By Lemma 10 then, after GREEDY_ROUTE2 has assigned routes to all the requests in $\sigma_{h,i}$, the link $(x_{h-1,2^i}, v_1)$ has congestion $i + h$.

Now notice that, by (25) and (26), $i = \log(n - 2d + 4) - 1$. Thus, $i + h = \log(n - 2d + 4) + d - 2$. On the other hand, by Lemma 11, an optimal algorithm can always find a set of edge disjoint routes with congestion 1. Therefore, the competitive ratio of GREEDY_ROUTE2 is $\Omega(d + \log(n - d))$. ■

We can also prove a result for GREEDY_ROUTE2 that is similar to that of Theorem 3 for GREEDY_ROUTE1. Specifically, we can prove the following theorem:

Theorem 9. The competitive ratio of GREEDY_ROUTE2 is $\Omega(d + \log((n/d) - d))$ on layered networks for arbitrarily large values of d and n .

This result implies a lower bound that is almost as high as that in Theorem 8 for the variation of GREEDY_ROUTE2 which breaks ties in favor of the shortest path.

4. CONCLUSIONS

We have given several results concerning the efficiency of greedy algorithms that assign routes to arbitrary sequences of permanent virtual circuit requests in an arbitrary communication network with equal capacity links. The simpler greedy algorithm, GREEDY_ROUTE1, was proposed by Mao and Simha [23]. We showed that the competitive ratio of GREEDY_ROUTE1 is $\Theta(\sqrt{\mathcal{D}m})$ on arbitrary networks when the bandwidth requirements of requests are approximately equal, where \mathcal{D} is the ratio of the longest-to-shortest path for any particular request and m is the number of network links. When bandwidth requirements are arbitrary, our upper bound is increased by a factor of $O(\sqrt{\mathcal{L}})$, where \mathcal{L} is the ratio of the maximum-to-minimum bandwidth requirement. We showed that a second greedy algorithm, GREEDY_ROUTE2, is superior to GREEDY_ROUTE1 when the length of the longest path in the network is small relative to the size of the network. Specifically, at least when the set of optimal routes for a request sequence has a small amount of overlap, GREEDY_ROUTE2 is $\max\{O(d \log n), O(\sqrt{\mathcal{D}m})\}$ competitive, where d is the length of the longest path assigned to a request and n is the number of network nodes. If $d = O(\log n)$, as is the case in many common networks, then the competitive ratio of GREEDY_ROUTE2 is polylogarithmic; if d is constant, it is asymptotically optimal. We also showed that the competitive ratio of GREEDY_ROUTE2 is $\Omega(d + \log(n - d))$ on arbitrary networks and $\Omega(d + \log((n/d) - d))$ on layered networks.

These results answer open questions posed by Mao and Simha [23] and also present alternatives to the asymptotically optimal, but computationally more expensive algorithm of Aspnes et al. [1, 2]. We discussed situations in which the greedy algorithms can be expected to perform well in Sections 2 and 3. Since there is a trade-off between speed and efficiency in the choice of algorithms, the decision of which to use really depends on the requirements of the situation. We would like to further investigate these questions in the future through simulations. It will be important to find appropriate networks for this purpose. One can always construct a network that makes the algorithms look good; we would ideally like to find real networks on which these algorithms would be appropriate.

There are a few technical points related to GREEDY_ROUTE1 and GREEDY_ROUTE2 that are topics of continuing research. First, we suspect that the $O(\sqrt{\mathcal{L}})$ term does not belong in the true competitive ratio of GREEDY_ROUTE1. In the proof, it appears that inequality (8) is weak; we would like to improve it to perhaps get a tighter result. More work also needs to be done on the competitive ratio of GREEDY_ROUTE2. While it was important to show that the competitive ratio of GREEDY_ROUTE2 is polylogarithmic for some cases, it is clear to us that this upper bound is not tight.

The two terms in the upper bound of Theorem 6 do not really “mesh;” the first term can be much larger than the second term for large values of d . Based on examples, we would conjecture that the competitive ratio of `GREEDY_ROUTE2` is closer to the lower bound of $O(d + \log n)$.

APPENDIX

The following simple lemma is used in the proof of Theorem 5 in Section 3:

Lemma 12. For any $a, y \geq 1$, $a \ln y \leq \lceil \log_{(a+1)/a} y \rceil \leq a \log_2 y + 1$.

Proof.

$$\begin{aligned}
 \log_{(a+1)/a} y &\leq \lceil \log_{(a+1)/a} y \rceil \leq \log_{(a+1)/a} y + 1 \\
 &\iff \\
 \frac{\ln y}{\ln\left(\frac{a+1}{a}\right)} &\leq \lceil \log_{(a+1)/a} y \rceil \leq \frac{\log_2 y}{\log_2\left(\frac{a+1}{a}\right)} + 1 \\
 &\iff \\
 \frac{a \ln y}{a \ln\left(\frac{a+1}{a}\right)} &\leq \lceil \log_{(a+1)/a} y \rceil \leq \frac{a \log_2 y}{a \log_2\left(\frac{a+1}{a}\right)} + 1 \\
 &\iff \\
 \frac{a \ln y}{\ln\left(\frac{a+1}{a}\right)^a} &\leq \lceil \log_{(a+1)/a} y \rceil \leq \frac{a \log_2 y}{\log_2\left(\frac{a+1}{a}\right)^a} + 1 \\
 &\iff \\
 \frac{a \ln y}{\ln e} &\leq \lceil \log_{(a+1)/a} y \rceil \leq \frac{a \log_2 y}{\log_2 2} + 1 \\
 &\iff \\
 a \ln y &\leq \lceil \log_{(a+1)/a} y \rceil \leq a \log_2 y + 1. \quad \blacksquare
 \end{aligned}$$

REFERENCES

- [1] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, “On-line load balancing with applications to machine scheduling and virtual circuit routing,” Proc ACM Symp on Theory of Computing, 1993, pp. 623–631.
- [2] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, On-line routing of virtual circuits with applications to load balancing and machine scheduling, J Assoc Comput Mach 44 (1997), 486–504.
- [3] B. Awerbuch and Y. Azar, “Local optimization of global objectives: Competitive distributed deadlock resolution and resource allocation,” Proc IEEE Symp on Foundations of Computer Science, 1994, pp. 240–249.
- [4] B. Awerbuch, Y. Azar, and A. Fiat, “Packet routing via min-cost circuit routing,” Proc Israeli Symp on the Theory of Computing and Systems, 1996, pp. 37–42.
- [5] B. Awerbuch, Y. Azar, and S. Plotkin, “Throughput-competitive on-line routing,” Proc IEEE Symp on Foundations of Computer Science, 1993, pp. 32–40.
- [6] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts, “Competitive routing of virtual circuits with unknown duration,” Proc ACM–SIAM Symp on Discrete Algorithms, 1994, pp. 321–327.
- [7] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosen, “Competitive non-preemptive call control,” Proc ACM–SIAM Symp on Discrete Algorithms, 1994, pp. 312–320.
- [8] B. Awerbuch, R. Gawlick, T. Leighton, and Y. Rabani, “On-line admission control and circuit routing for high performance computing and communication,” Proc IEEE Symp on Foundations of Computer Science, 1994, pp. 412–423.
- [9] Y. Azar, B. Kalyanasundaram, S. Plotkin, K.R. Pruhs, and O. Waarts, “On-line load balancing of temporary tasks,” Workshop on Algorithms and Data Structures, 1993, pp. 119–130.
- [10] Y. Azar, J. Naor, and R. Rom, “The competitiveness of on-line assignments,” Proc ACM–SIAM Symp on Discrete Algorithms, 1992, pp. 203–210.
- [11] Y. Azar, J. Naor, and R. Rom, The competitiveness of on-line assignments, J Alg 18 (1995), 221–237.
- [12] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra, “New algorithms for an ancient scheduling problem,” Proc ACM Symp on Theory of Computing, 1992, pp. 51–58.
- [13] J.A. Garay and I.S. Gopal, “Call preemption in communication networks,” Proc IEEE INFOCOM, 1992, pp. 1043–1050.
- [14] R. Gawlick, C. Kalmanek, and K.G. Ramakrishnan, “On-line routing for permanent virtual circuits,” IEEE INFOCOM, 1995, pp. 278–288.
- [15] R.L. Graham, Bounds for certain multiprocessing anomalies, Bell Syst Tech J 45 (1966), 1563–1581.
- [16] J.T. Havill and W. Mao, “Greedy on-line file transfer routing,” Proc IASTED International Conf on Parallel and Distributed Systems, 1997, pp. 225–230.
- [17] J.T. Havill, W. Mao, and R. Simha, “A lower bound for on-line file transfer routing and scheduling,” Proc 31st Annual Conf on Information Sciences and Systems, 1997, pp. 936–941.
- [18] D.R. Karger, S.J. Phillips, and E. Torng, “A better algorithm for an ancient scheduling problem,” Proc ACM–SIAM Symp on Discrete Algorithms, 1994, pp. 132–140.
- [19] R.M. Karp, On-line algorithms versus off-line algorithms: How much is it worth to know the future? Technical Report TR-92-044, International Computer Science Institute, 1992.
- [20] P. Klein, S. Plotkin, C. Stein, and E. Tardos, Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and

- finding sparse cuts, *SIAM J Comput* 23 (1994), 466–487.
- [21] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos, and S. Tragoudas, “Fast approximation algorithms for multicommodity flow problems,” *Proc ACM Symp on Theory of Computing*, 1991, 101–111.
- [22] S. Leonardi, A. Marchetti-Spaccamela, A. Presciutti, and A. Rosen, “On-line randomized call control revisited,” *Proc ACM–SIAM Symp on Discrete Algorithms*, 1998, 323–332.
- [23] W. Mao and R. Simha, Routing and scheduling file transfers in packet-switched networks, *J Comput Inf* 1 (1994), 559–574.
- [24] S. Plotkin. Competitive routing of virtual circuits in ATM networks, *IEEE J Select Areas Commun* 13 (1995), 1128–1136.
- [25] F. Shahrokhi and D.W. Matula, The maximum concurrent flow problem, *J Assoc Comput Machin* 37 (1990), 318–334.