Online Client-Server Load Balancing Without Global Information

Baruch Awerbuch[*] *Mohammad T. Hajiaghayi*[†] *Robert D. Kleinberg*[†‡§] *Tom Leighton*[‡]

Abstract

We consider distributed online algorithms for maximizing throughput in a network of clients and servers, modeled as a bipartite graph. Unlike most prior work on online load balancing, we do not assume centralized control and seek algorithms and lower bounds for decentralized algorithms in which each participant has only local knowledge about the state of itself and its neighbors. Our problem can be seen as analogous to the recent work on oblivious routing in [8, 14, 19], but with the objective of maximizing throughput rather than minimizing congestion. In contrast to that work, we prove a strong lower bound (polynomial in n, the size of the graph) on the competitive ratio of any oblivious algorithm. This is accompanied by simple algorithms achieving upper bounds which are tight in terms of k, the maximum throughput achievable by an omniscient algorithm. Finally, we examine a restricted model in which clients, upon becoming active, must remain so for at least $\log(n)$ time steps. In contrast to the primarily negative results in the oblivious case, here we present an algorithm which is constant-competitive. Our lower bounds justify the intuition, implicit in earlier work on the subject [2], that some such restriction (i.e. requiring some stability in the demand pattern over time) is necessary in order to achieve a constant — or even polylogarithmic — competitive ratio.

1 Introduction

We consider distributed online algorithms for maximizing throughput in a network of clients and servers, modeled as a bipartite graph $G = (V_L, V_R, E)$ with V_L representing the clients, V_R representing the servers, and E representing the client-server assignments which are considered admissible, e.g. because of proximity constraints. Motivated by Internet load-balancing applications, such as load-balancing HTTP connections in a content delivery network, we consider the case where client-server connections are extremely short-lived (lasting for only one unit of time) and it is impossible to get an instantaneous snapshot of the demand pattern. Our focus is on distributed algorithms in which clients must make decisions knowing nothing about the current demand pattern other than their own demand, and servers must make decisions knowing nothing other than what they learn from their adjacent clients. (We also assume that servers may report their load to the adjacent clients at the end of a round, though this is not necessarily predictive of their load in future rounds.) This emphasis on distributed algorithms with a very limited amount of communication is what distinguishes the present paper from most of the previous work on online load balancing.

Our model of online load-balancing is completely stateless, since client-server connections last for only one unit of time and the demand pattern may be completely different in the next round. Thus, if the demand pattern is allowed to vary arbitrarily and adversarially over time, an online algorithm's competitive ratio over a sequence of T steps will generally be no better than its competitive ratio in the case $T = 1$, i.e. a one-shot game between the algorithm and the adversary. It may seem hopeless to achieve non-trivial upper bounds on competitive ratio in such a one-shot game, since the algorithm has no time to learn any information about the demand pattern before making its decisions. However, the recent result of Räcke [19] (and its subsequent constructive results by Harrelson, Hildrum, and Rao [14] and Bienkowski, Korzeniowski and Räcke [8]) on oblivious routing in undirected networks demonstrate that it is sometimes possible to achieve surprisingly strong upper bounds for such "one-shot" load balancing problems. Specifically, for a multicommodity flow problem in an undirected graph G, it is possible for each commodity to choose a flow *without knowing the demand of any other commodity*, in such a way that the maximum edge congestion in G is within a $O(\log^2 n \log\log n)$ factor of that of the congestion-minimizing flow for the given demand pattern.

Much of the present paper is devoted to analyzing the analogous question for client-server load-balancing. While it is possible to achieve competitive ratios significantly better than the trivial $O(n)$ bound for this problem, we show that it is impossible to achieve a $\mathrm{polylog}(n)$ competitive ratio. A comparably strong lower bound for oblivious routing in bipartite directed graphs was established using a simple construction in [6]. Our lower bound requires a significantly more sophisticated construction because we seek a lower bound on competitive ratio for *throughput* rather than edge congestion. This is the first polynomial lower bound on throughput for oblivious routing.

[*]Center for Networking and Distributed Systems, Johns Hopkins University, U.S.A., {baruch@cs.jhu.edu}

[†]Department of Mathematics and Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 200 Technology Square, Cambridge, MA 02139, U.S.A., {hajiagha,rdk,ftl}@theory.lcs.mit.edu

[‡]Akamai Technologies, 8 Cambridge Center, Cambridge, MA 02139, U.S.A.

[§]Supported by a Fannie and John Hertz Foundation Fellowship.

As a counterpoint to these primarily negative results, we consider a restricted adversarial model in which clients have $\{0,1\}$-valued demands (i.e. they are either active or inactive), and a client who becomes active must remain active for at least r rounds thereafter. In this environment, we present an algorithm whose competitive ratio is $O(\Delta^{6/r})$, where Δ is an upper bound (known to all parties) on the degree of any client. In particular, the algorithm achieves a constant competitive ratio when $r = \Omega(\log \Delta)$. Our algorithm is structurally similar to the concurrent routing algorithm of [2], with two important differences: the latter algorithm assumes that clients are not entering and leaving the system over time, and it requires the clients to gradually increase their flow until eventually reaching the desired level of throughput. Our algorithm permits clients to become active and inactive over time (provided that a client, upon becoming active, remains active for the next r steps), and it permits them to route their full demand in each round in which they are active (though the demand may not be satisfied, if it is sent to a congested server).

All of the algorithms presented in this paper are very easy to implement, requiring surprisingly straightforward decision-making and communication protocols on the part of clients and servers. Some of the lower bound proofs, on the other hand, are relatively sophisticated. We show that oblivious algorithms for throughput maximization can be obstructed by the presence of substructures in the bipartite graph which we call γ-focal matchings. The task of proving competitive-ratio lower bounds is thereby reduced to a combinatorial problem of packing as many γ-focal matchings as possible into a bipartite graph of size n. Our construction of such graphs involves an interesting mixture of combinatorial, algebraic, and probabilistic techniques. These lower bound techniques constitute one of main contributions of this paper, and we believe it may be interesting to consider whether they can be used to obtain lower bounds for other applications.

2 Related work

Recall that this paper considers load balancing for a client-server model which has two essential characteristics. The first one is that our system is fully dynamic and the input can change drastically from one time to the next time. The second one is that there is no central "dispatcher" in the system that could communicate the result of the maximum matching computation to the clients, thus guiding their routing decisions. Indeed, the interplay of these two aspects plays an important role in this paper, since otherwise there are many algorithms in the literature for models possessing only one of these characteristics. Below, we review some of these results.

2.1 Centralized control Finding a maximum matching, or its generalization to maximum flow, is one of the classical problems in combinatorial optimization. The fastest known sequential algorithm for the problem has running time close to $O(|E||V|)$ [12]. For the more general problem of solving a positive linear program to within a $(1+\epsilon)$ factor of optimality, Plotkin, Shymoys and Tardos [18] present a sequential algorithm which repeatedly identifies a globally minimum weight path, and pushes more flow along that path. The algorithm of Plotkin et al. is further improved by Garg and Konemann [10], who give faster and simpler primal-dual algorithms for multicommodity flow and other fractional packing problem with the same approximation factor $(1+\epsilon)$. In addition, several (deterministic and randomized) parallel algorithms for maximum bipartite matching and maximum flow have been proposed (see e.g. [9, 12, 15]). Although, these algorithms have efficient implementation, they are all centralized algorithms and require global kowledge of the demand pattern and global coordination, which make them unsuitable for fast distributed implementation with local information.

2.2 Distributed control with persistent demands Routing and admission control. Assuming that the demand pattern remains stable for at least $\Omega(\log n)$ rounds at a time, a distributed routing and flow control algorithm with a global objective function has been given by Awerbuch and Azar [2]. This work is based on fundamental results from competitive analysis [1, 3] and assumes clients can gradually increase their flow; while the flow is still small it could for example be buffered at the client. In this case, under the assumption that there is a small number of routing paths, they provide an $O(\log n)$ competitive algorithm for the routing problem, which takes a polylogarithmic number of rounds to converge. Awerbuch and Leighton [4, 5] have suggested general methods for distributed routing and admission control that use a polynomial amount of buffer space. Our lower bounds demonstrate that at least one of these two assumptions — persistence of demands over time, or the ability to buffer packets —is really required in order to achieve a polylogarithmic competitive ratio.

Distributed admission control alone. For the distributed admission control problem (in which clients do not choose a server or routing path, but only their sending rate) Papadimitriou and Yannakakis [17] initiated the study of flow control using distributed routers based only on local information. More precisely, they presented a framework for solving positive linear programs by distributed agents. Luby and Nisan [16], Bartal, Byers and Raz [7] and finally Garg and Young [11] obtained a $(1+\epsilon)$ competitive algorithms converging in a polylogarithmic number of rounds.

Even though all of these results are distributed, they converge to their final solution in a polylogarithmic number of rounds, which makes them unsuitable for our fully dynamic client-server model.

2.3 Distributed control without persistence of demands One possible approach to distributed load-balancing is to use an "oblivious" solution. Such an oblivious algorithm exists

for the congestion minimization problem in undirected edge-capacitated graphs ([19] and its subsequent improvement by Harrelson, Hildrum, and Rao [14]) and for directed and node-capacitated graphs [13]. No such solution exists for the throughput problem, though Räcke and Rosen (independently and concurrently with our work) gave a distributed online call control algorithm which is closely related to oblivious throughput maximization in undirected graphs [20]. One of the main results in our paper establishes nearly tight upper and lower bounds on the performance of oblivious routing schemes in *directed bipartite graphs*, in terms of throughput. The performance gap between the optimal and oblivious solution is polynomial; our lower bounds show that this gap is inherent.

3 Formal model and statement of results

Our graph terminology is as follows. All the graphs in this paper are directed bipartite graphs without multiple edges. For such a graph $G = (V_L, V_R, E)$, we will refer to elements of V_L as *clients* and elements of V_R as *servers*. The number of clients is denoted by n, the number of servers by m. The edges of E are directed from clients to servers. For a vertex set $S \subseteq V_L \cup V_R$ we denote the set of adjacent vertices by $\Gamma(S)$, the set of outgoing edges by $\delta^+(S)$, and the set of incoming edges by $\delta^-(S)$. When S is a singleton set $\{v\}$, these will be abbreviated to $\Gamma(v), \delta^+(v), \delta^-(v)$. The degree of a vertex v will be denoted by $d(v)$.

The prototypical problem we will analyze is the following online throughput maximization problem. In each time step t $(1 \le t \le T)$, an adversary designates a set S_t of clients, called the *active clients*. Each active client i generates a request and must choose a (possibly random) adjacent server to which it will send this request, without knowing which other clients are active. Each server that receives one or more requests in round t may choose to satisfy any one of them. The goal is to maximize the expected number of satisfied requests, called the *throughput*. The algorithm is judged according to its competitive ratio, i.e. the ratio of its throughput to that of the omniscient algorithm which chooses a throughput-maximizing assignment in each period.

When the problem is posed in this form, its online nature is essentially irrelevant. This is because any algorithm achieving the optimum competitive ratio in the $T = 1$ case also achieves the optimum competitive ratio in the case of general T, by simply ignoring past history and treating each round as if it were the first round. For this reason, we will focus most of our attention on the $T = 1$ case, which we call the *one-shot model*. We will use the letter k to denote the throughput of the optimal assignment, i.e. the size of a maximum matching from the active clients to V_R. The following variants of the problem are also of interest.

Multicast model. In contrast to the *unicast model* described above, we may consider a model in which a client may send its request to any subset of the adjacent servers. A server receiving one or more requests may choose to satisfy any one of them, but it must make this choice without any knowledge about the set of active clients other than the information contained in the requests it received. The throughput is defined as the number of *distinct* clients whose requests are satisfied, i.e. a client whose request is satisfied by two or more servers still contributes only 1 to the throughput.

Fractional assignments. Instead of requiring each active client i to choose one of its adjacent servers, it may choose a fractional load distribution among its outgoing edges. In other words, each client chooses a function $f_i : \delta^+(i) \to [0,1]$ satisfying $\sum_{e \in \delta^+(i)} f(e) \le 1$. As always, client i must specify f_i without knowing which other clients are in S. The load on a server j, denoted by $\ell(j)$, is equal to the total load on all incoming edges. The throughput is defined by $\sum_{j \in V_R} \min\{1, \ell(j)\}$.

Restricted adversary. In the restricted-adversary model, we assume that the sets S_t of active clients satisfy the following constraint: every client, upon becoming active, must remain active for the next r rounds. In other words, if $i \in S_t$ then there exist t_0, t_1 such that $t_0 \le t \le t_1$, $t_1 - t_0 \ge r$, and $i \in S_{t'}$ for $t_0 \le t' \le t_1$. (We call r the *minimum activity period*.) We also assume that servers may report their load and capacity to the adjacent clients at the end of each round.

In proving lower bounds in this paper, we will assume that the structure of the entire graph G is known to all clients and servers, and that they have access to an unlimited supply of shared random bits. In contrast, our upper bounds will be based on algorithms which require much less knowledge on the part of the participants: each vertex need only know which vertices are adjacent to it. (In Section 7 we must also assume that they share a common estimate of the maximum client degree, Δ.)

The following theorems summarize our main results.

THEOREM 3.1. *In the unicast one-shot model, there is an algorithm whose competitive ratio is $O(\sqrt{k})$, and this bound is tight in terms of k, even if the algorithm is randomized and is allowed to use fractional assignments. In terms of n, the competitive ratio of any such algorithm is $\Omega(n^{0.103})$.*

THEOREM 3.2. *In the multicast one-shot model, there is an algorithm whose competitive ratio is $O(k^{1/3})$, provided that the servers know the degree of their adjacent clients or that the clients can communicate this information in their request headers. This bound is tight in terms of k, even if the clients are allowed to put an arbitrary amount of information in the request header. In terms of n, the competitive ratio of any such algorithm is $\Omega(n^{0.069})$.*

THEOREM 3.3. *In the restricted adversary model with fractional assignments and with minimum activity period r, if the clients know the value of r as well as an upper bound Δ on the maximum degree of any client, then there is an algorithm whose competitive ratio is $O(\Delta^{6/r})$. In particular, if $r = \Omega(\log \Delta)$, the competitive ratio is constant.*

It is worth mentioning that the proofs of Theorems 3.1 and 3.2 also establish tight bounds on the optimal competitive ratio in terms of m, the number of servers. The optimal competitive ratio is $\theta(m^{1/2})$ in the unicast model and $\theta(m^{1/3})$ in the multicast model. Tightening the bounds in terms of n remains an open question.

4 Lower bounds for the one-shot model

Our lower bounds in the one-shot model depend on finding matchings M between a set of clients M_L and servers M_R, such that removing M from the edge set of G leaves M_L with a very small set of neighbors. We call such structures γ-focal matchings; the precise definition is given below. When G contains a γ-focal matching and the set of active clients is M_L, it is important for most active clients to send their request along an edge of M, since all other edges go through a very limited number of servers, leading to very limited throughput. Suppose an active client has many outgoing edges, and each of them belongs to a different γ-focal matching. If the set of active clients is equal to M_L for one of these matchings M, but it is not known which matching, then such a client is unlikely to send its request along the outgoing edge which belongs to M, and consequently it is impossible for the algorithm to achieve high expected throughput.

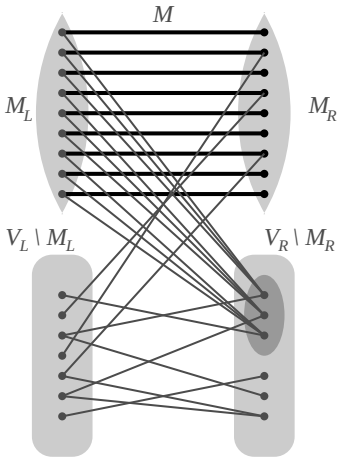

Figure 1: A γ-focal matching.

DEFINITION 4.1. *Let M be a matching in G, and let M_L, M_R denote the sets of left and right endpoints, respectively, of the matching edges. We call M a* γ-focal matching *if $|\Gamma(M_L)\setminus M_R| < |M|/\gamma$ and G contains no edges between M_L and M_R other than those which belong to M.*

4.1 Unicast lower bounds Let $\mathcal{A}$ denote the set of all *fractional assignments* in G, i.e. $\mathcal{A} = \{f : E \to [0,1] \mid \forall i \in V_L \sum_{j\in\Gamma(i)} f(i,j) = 1\}$. A set S of active clients may be represented by a function $D : V_L \to \{0,1\}$ mapping S to 1 and $V_L \setminus S$ to 0; we call this the *demand pattern* associated with S. Given a fractional assignment f, define the load on server j by $\ell(j) = \sum_{i\in\Gamma(j)} f(i,j)D(i)$ and the throughput of f by $\theta(f) = \sum_{j\in V_R} \min\{\ell(j), 1\}$. We may think of a randomized assignment algorithm in the one-shot model as computing a function $A : \{0,1\}^{V_L} \times X \to \mathcal{A}$, where X is a probability space encapsulating all the random bits (both shared and private) which the parties may use in their decision-making. The fact that the assignment is *oblivious* (i.e. that clients must choose their own assignment without knowing which other clients are active) is captured by the following constraint: in the fractional assignment $f = A(D,x)$, for any edge (i,j), the value of $f(i,j)$ may only depend on $D(i)$ and x. In other words, if $f' = A(D',x)$ and $D'(i) = D(i)$, then $f'(i,j) = f(i,j)$.

THEOREM 4.1. *Let G be a bipartite graph which is (d_L, d_R)-biregular, i.e. every $i \in V_L$ has degree d_L and every $j \in V_R$ has degree d_R. Assume all servers have unit capacity. If the edge set of G can be partitioned into γ-focal matchings of equal size, then the competitive ratio of any oblivious randomized fractional assignment algorithm for G is at least $\frac{1}{2}\min\{d_L, \gamma\}$.*

Proof. Let A be any oblivious randomized fractional assignment algorithm, and let $M^{(1)}, \ldots, M^{(s)}$ be a partition of E into γ-focal matchings of equal size k. Note that the number of edges satisfies $sk = |E| = d_L n$, whence $\frac{n}{s} = \frac{k}{d_L}$. Let the demand pattern $D : V_L \to \{0,1\}$ be defined by selecting a matching $M = M^{(r)}$ uniformly at random from $\{M^{(1)}, \ldots, M^{(s)}\}$, independently of the algorithm's random seed $x \in X$, and setting $D(i) = 1$ if $i \in M_L$, 0 otherwise. The throughput of the assignment $f = A(D,x)$ satisfies

$$\begin{aligned}\theta(f) &= \sum_{j\in M_R} \min\{\ell(j),1\} + \sum_{j\in\Gamma(M_L)\setminus M_R} \min\{\ell(j),1\} \\ &\le \sum_{j\in M_R} \ell(j) + \sum_{j\in\Gamma(M_L)\setminus M_R} 1 \le \sum_{e\in M} f(e) + k/\gamma.\end{aligned}$$

Let D^* denote the demand pattern in which all clients are active, i.e. $D^*(i) = 1\,\forall i$, and let $f^* = A(D^*, x)$. By the definition of "oblivious", we have that $f(i,j) = f^*(i,j)$ for all $i \in M_L$. Hence $\theta(f) \le k/\gamma + \sum_{e\in M} f^*(e)$. Now let's take the expectation over the random choice of x and M.

$$\begin{aligned}\mathbf{E}[\theta(f)] &\le \frac{k}{\gamma} + \sum_{e\in E} \Pr(e \in M)\mathbf{E}[f^*(e)] = \frac{k}{\gamma} + \frac{1}{s}\sum_{e\in E}\mathbf{E}[f^*(e)] \\ &= \frac{k}{\gamma} + \frac{1}{s}\mathbf{E}\left[\sum_{i\in V_L}\sum_{e\in\delta^+(i)} f^*(e)\right] \le \frac{k}{\gamma} + \frac{n}{s} = \frac{k}{\gamma} + \frac{k}{d_L} \\ &\le \frac{2k}{\min\{d_L,\gamma\}}.\end{aligned}$$

The optimal assignment sends the k requests along the edges of M, thus achieving throughput k. Hence the competitive ratio of A is at least $\frac{1}{2}\min\{d_L, \gamma\}$, as claimed.

THEOREM 4.2. *There exists a bipartite graph G such that the competitive ratio of any oblivious randomized fractional assignment algorithm for G is at least $\sqrt{k}/2$, where k is the maximum throughput achievable in the given problem instance.*

Proof. The graph G is defined as follows. Given a positive integer d, let V_R be the set $\{1, 2, \ldots, d^2\}$, and let V_L be the set of all d-element subsets of V_R. Each such set $i \in V_L$ is joined by an edge to each of its elements $j \in V_R$. G is a biregular bipartite graph, with $d_L = d$ and $d_R = \binom{d^2-1}{d-1}$.

For each $(d-1)$-element subset $S \subseteq V_R$, let $M(S)$ be the matching containing, for each $j \in V_R \setminus S$, an edge from $i = S \cup \{j\}$ to j. Each such matching has size $d^2 - d + 1$, and each edge $(i,j) \in E$ belongs to exactly one such matching $M(S)$. (Namely, $S = i \setminus \{j\}$.)

We claim that each matching $M = M(S)$ is a $(d-1)$-focal matching. We have $M_R = V_R \setminus S$, and each $i \in M_L$ has one edge joining it to M_R (namely, the matching edge) and $d-1$ edges joining it to S. Thus G contains no edges between M_L and M_R other than the matching edges, and

$$|\Gamma(M_L) \setminus M_R| \leq |M|/d,$$

because the left side is equal to $d-1$ while the right side is equal to $d - 1 + 1/d$.

Applying Theorem 4.1, we find that the competitive ratio of any oblivious randomized fractional assignment algorithm is at least $d/2$, which is greater than $\sqrt{k}/2$ since $k = d^2 - d + 1$.

Note that the proof of Theorem 4.2 also gives a lower bound of $\sqrt{m}/2$ where m is the number of servers. (We will see later that this bound is tight in terms of m.) However the number of clients in this example, n, is equal to $\binom{d^2}{d}$, so the competitive-ratio lower bound of $d/2$ only translates into a very weak lower bound of $\Omega(\log n / \log\log n)$ in terms of n. The following theorem demonstrates that a much stronger lower bound is possible.

THEOREM 4.3. *There exists a bipartite graph G such that the competitive ratio of any oblivious randomized fractional assignment algorithm for G is $\Omega(n^{0.103})$.*

Proof. The construction of the graph G in this case is quite complicated. For a positive integer d, let X be the ring $(\mathbb{F}_2)^d$, i.e. the cartesian product of d copies of the field $\mathbb{F}_2 = \{0,1\}$. Considering X as a vector space over $\mathbb{F}_2$, let Y be a linear subspace of dimension bd. (We will optimize the value of the parameter $b < 1$ later on.) Let Z denote the set of all z in X such that zy is non-zero for all non-zero $y \in Y$. (If we identify elements of X with subsets of $\{1, 2, \ldots, d\}$, then the non-zero elements of Y constitute a set system and Z consists of all hitting sets for this set system.) We will want the complement, $X \setminus Z$, to be as small as possible. Here is a calculation which bounds the expected size of $X \setminus Z$ when Y is a *random* linear subspace of dimension bd. For any non-zero $y \in X$, the probability that it belongs to Y is $2^{(b-1)d}$, and the number of z such that $zy = 0$ is $2^{d - wt(y)}$, where $wt(y)$ denotes the Hamming weight of y. This means that an upper bound for the expected size of $X \setminus Z$ is given by:

$$\begin{aligned}\sum_{y \in X, y \neq 0} 2^{(b-1)d} 2^{d-wt(y)} &= 2^{bd} \sum_{j=1}^{d} \binom{d}{j} 2^{-j} \\ &= 2^{bd}\left[(3/2)^d - 1\right] < (3 \cdot 2^{b-1})^d\end{aligned}$$

Henceforth we assume that we have chosen a specific linear subspace Y such that the cardinality of $X \setminus Z$ is at most $(3 \cdot 2^{b-1})^d$. Later on, when we specify the value of b, it will be the case that $3 \cdot 2^{b-1} = \sqrt{3} \cdot (1 + o(1))$, so the fraction of elements of X which are not contained in Z is exponentially small in d.

The bipartite graph G is defined as follows. We put $V_L = X \times Z$, $V_R = X$, and $E = \{((x_L, z_L), x_R) \mid x_R - x_L = yz_L \text{ for some } y \in Y\}$. By abuse of notation, we will write an edge e with left endpoint (x_L, z_L) and right endpoint x_R as an ordered triple $e = (x_L, z_L, x_R)$. Note that if $x_R - x_L = yz_L$ for some $y \in Y$, then this y is actually unique. (If $yz_L = y'z_L$, then $(y - y')z_L = 0$. Since $y - y' \in Y$ and $z \in Z$, this implies $y - y' = 0$.) We will refer to this unique value of y as the *type* of edge $e = (x_L, z_L, x_R)$.

This proves that each $(x_L, z_L) \in V_L$ has exactly $|Y|$ outgoing edges, one of each type $y \in Y$. Similarly, each $x_R \in V_R$ has exactly $|Y \times Z|$ incoming edges. Given $(y, z) \in Y \times Z$, one may easily verify that there is one and only one edge of type y joining $X \times \{z\}$ to x_R, namely the edge $e = (x_R - yz, z, x_R)$.

This establishes that G is $(2^{bd}, 2^{bd} \cdot |Z|)$-biregular. We must now specify a partition of the edge set into γ-focal matchings of equal size. For each pair (x, y) where $x \in X$, $y \in Y$, let

$$M(x,y) = \{(x + ((1-y)z), z, x + z) \mid z \in Z\}.$$

where "1" denotes the vector $(1, 1, 1, \ldots, 1) \in X$. Note that $(x + ((1-y)z), z, x + z)$ is a valid edge of type y in G, because $x + z = x + 1 \cdot z = x + ((1-y)z) + yz$. The matchings $M(x,y)$ each have size $|Z|$. To see that every edge belongs to exactly one such matching, observe that if $e = (x_L, z_L, x_R)$ with $x_R - x_L = yz_L$, then $e \in M(x_R - z_L, y)$. There can be no other $M(x', y')$ containing e, since y' must equal the type of e and x' must equal $x_R - z_L$ in order for e to belong to $M(x', y')$.

Next, we wish to see that each such matching $M = M(x,y)$ is γ-focal for a reasonably large (i.e. exponential in d) value of γ. To do so, we will first show that every edge between M_L and the set $M_R = x + Z = \{x + z \mid z \in Z\}$ belongs to M. Let $e = (x_L, z_L, x_R)$ be such an edge, with $x_R - x_L = y'z_L$ for some $y' \in Y$. Since $(x_L, z_L) \in M_L$, we have $x_L = x + (1-y)z_L$, whence $x_R = x + (1 + y' - y)z_L$. If $y' = y$ then $e \in M$. If $y' \neq y$, then we use the fact that every element w of the ring X satisfies $w(1-w) = 0$. Applying this with $w = y - y'$, we see that $(y - y')(1 + y' - y)z_L = 0$. As $y - y'$ is a non-zero element of Y, we may conclude that $(1 + y' - y)z_L \notin Z$, whence $x_R \notin x + Z$. Finally, observe that

$$|\Gamma(M_L) \setminus M_R| \leq |V_R \setminus M_R| = |X \setminus (x + Z)| \leq (3 \cdot 2^{b-1})^d.$$

Recalling that $|M| = |Z| = (1 - o(1))2^d$, we see that M is γ-focal with $\gamma = (1 - o(1))\left(2^{2-b}/3\right)^d$.

Applying Theorem 4.1, we find that no oblivious randomized fractional assignment algorithm achieves a competitive ratio better than

$$\frac{1}{2}\min\{d_L, \gamma\} = \frac{1}{2}\min\left\{2^{bd}, \left(2^{2-b}/3\right)^d (1 - o(1))\right\}.$$

This approximately maximized when $2^b = 2^{2-b}/3$, i.e. when $b = 1 - \frac{1}{2}\log_2(3) = 0.2075\ldots$ (Of course, b must be rounded to the nearest multiple of $1/d$, since bd, the dimension of the vector space Y, must be an integer.) Recalling that $n = |X \times Z| < 2^{2d}$, we see that the lower bound of $\Omega(2^{bd})$ on competitive ratio may be expressed as $\Omega(n^{b/2}) = \Omega(n^{0.103})$.

4.2 Multicast lower bounds Proving lower bounds in the multicast model is slightly more difficult than in the unicast model, because clients may broadcast their request to every adjacent server if they wish. If the set of active clients is equal to M_L for some γ-focal matching M, and

each client chooses to broadcast its request to all adjacent servers, then each server in M_R will receive exactly one request and will satisfy it, leading to a throughput of $|M|$, the optimum throughput for the designated set of active clients. Nevertheless, it is possible to use γ-focal matchings to prove lower bounds in the multicast model, by combining them with another device which we call a *smokescreen*. A smokescreen is simply a random set of clients whose size is small relative to the size of the matching, and whose purpose is to confuse the servers in M_R by making it difficult for them to distinguish which incoming request is coming from M_L. These techniques enable us to establish the following theorem.

THEOREM 4.4. *Let G be a bipartite graph which is (d_L, d_R)-biregular. If the edge set of G can be partitioned into γ-focal matchings of size $k = \Omega(m)$, then the competitive ratio of any oblivious assignment protocol for G in the multicast model is $\Omega(\min\{\gamma, \sqrt{d_L}\})$.*

Proof. We will begin by formalizing the class of protocols which we will be considering. We will assume once again that there is a probability space X encapsulating the random bits (both shared and private) available to the parties in their computation. There is also a (not necessarily finite) message space M encapsulating all the messages that clients may send to servers. A protocol is specified by a communication function $A_i : \{0,1\} \times X \to M^{d(i)}$ for each client i and a decision function $B_j : M^{d(j)} \times X \to \Gamma(j)$ for each server j. The value of $A_i(D, x)$ specifies the $d(i)$-tuple of messages which i will send on its outgoing edges if its demand is D and the random seed is x. The value of $B_j(m_1, m_2, \ldots, m_{d(j)}, x)$ specifies which client's request will be served by j if the random seed is x and j receives messages $m_1, m_2, \ldots, m_{d(j)}$ on its incoming edges. We will call such a protocol $\{A_i, B_j\}$ an *oblivious assignment protocol for G in the multicast model*.

Without loss of generality we may assume that $M = \{0,1\}$ and that each communication function A_i is simply the function $A_i(D, x) = D$. In other words, each client simply informs all adjacent servers whether it is active or not. This assumption is without loss of generality because for any other protocol $\hat{\mathcal{P}} = \{\hat{A}_i, \hat{B}_j\}$, we can construct a protocol $\mathcal{P} = \{A_i, B_j\}$ with A_i defined as above, and with B_j defined as follows. For each client $i \in \Gamma(j)$, $B_j(m_1, \ldots, m_{d(j)}, x)$ simulates $\hat{A}_i(m_i, x)$ to determine what message $\hat{m}_i$ would have been sent from i to j under the protocol $\mathcal{P}$, and it then computes $\hat{B}_j(\hat{m}_1, \ldots, \hat{m}_{d(j)}, x)$ to determine what request it would have satisfied. This new protocol $\mathcal{P}$ has precisely the same outcome as $\hat{\mathcal{P}}$. Based on this reduction, we will assume from now on that each server's decision function is a mapping $B_j : \{0,1\}^{\Gamma(j)} \times X \to \Gamma(j)$ which chooses, for each subset $S \subseteq \Gamma(j)$, a random element $B_j(S, x) \in \Gamma(j)$ determined by the random seed x. The notion that servers have difficulty distinguishing elements of M_L from elements of the smokescreen is captured by the following easy lemma, whose proof is deferred to the full version of the paper.

LEMMA 4.1. *Let Γ be a set of d elements, and consider any function $B : 2^\Gamma \to \Gamma$. Suppose a random element $i \in \Gamma$ is sampled according to the uniform distribution, and a random set $S \subseteq \Gamma \setminus \{i\}$ is sampled by choosing each element independently with probability p. Then $\Pr(B(S \cup \{i\}) = i) = O\left(\frac{1}{pd}\right)$.*

Now to prove Theorem 4.4, let $M^{(1)}, \ldots, M^{(s)}$ be a partition of the edge set into γ-focal matchings of size k, and let the set of active clients S be defined as follows. Every client $i \in M_L$ belongs to S, and in addition, every $i \in V_L \setminus M_L$ joins S independently with probability $p = \sqrt{\frac{m}{d_R n}}$. The set $Q = S \setminus M_L$ are referred to as the *smokescreen*.

In the discussion preceding Lemma 4.1, we argued that one can assume w.l.o.g. that the protocol operates as follows: each client broadcasts its request to all adjacent servers; each server j receives requests from a set $T_j \subseteq \Gamma(j)$ and chooses which request to satisfy by computing a function $B_j(T_j, x)$ which depends on T_j and the (shared) random seed x.

Since each client in Q and each server in $\Gamma(M_L) \setminus M_R$ contributes at most one unit of throughput, we have the following bound on the expected total throughput θ (where the expectation is over the random choice of S as well as the random seed x):

$$\begin{aligned}\theta &\leq \mathbf{E}[|Q|] + \mathbf{E}[|\Gamma(M_L) \setminus M_R|] \\ &\quad + \sum_{j \in V_R} \Pr(j \in M_R \wedge B_j(T_j, x) \in M_L) \\ &\leq pn + k/\gamma + \sum_{j \in V_R} \Pr(B_j(T_j, x) \in M_L \| j \in M_R).\end{aligned}$$

We may bound $\Pr(B_j(T_j, x) \in M_L \| j \in M_R)$ using Lemma 4.1. The key observation is that, conditional on the event $j \in M_R$, the set of active clients adjacent to j consists of one element i of M_L, uniformly distributed in $\Gamma(j)$, as well as a random subset of $\Gamma(j) \setminus \{i\}$ sampled by including each element independently with probability p. Thus

$$\Pr(B_j(T_j, x) \in M_L \| j \in M_R) = O\left(\frac{1}{pd_R}\right) = O\left(\sqrt{\frac{1}{d_L}}\right),$$

where the last step follows from the fact that $md_R = |E| = nd_L$. We are assuming $k = \Omega(m)$, so

$$\begin{aligned}\theta &\leq pn + k/\gamma + O(m\sqrt{1/d_L}) \\ \theta/k &\leq O\left(\frac{pn}{m} + \frac{1}{\gamma} + \sqrt{\frac{1}{d_L}}\right) = O\left(\max\left\{\frac{1}{\gamma}, \sqrt{\frac{1}{d_L}}\right\}\right),\end{aligned}$$

and the competitive ratio k/θ is $\Omega(\min\{\gamma, \sqrt{d_L}\})$.

THEOREM 4.5. *There exists a graph G such that the competitive ratio of any oblivious assignment protocol for G in the multicast model is $\Omega(k^{1/3})$.*

Proof. For a positive integer d, let $V_R = \{1, 2, \ldots, d^3\}$, let V_L be the set of all d^2-element subsets of V_R, and define the edge set by joining such a set i to an element $j \in V_R$ if j is an element of i, as in the proof of Theorem 4.2. As in that proof, the edge set may be partitioned into $(d-1)$-focal matchings; these matchings have size $k = d^3 - d^2 + 1 = \Omega(m)$. We have $\gamma = d - 1 = \Omega(k^{1/3})$ and $\sqrt{d_L} = d = \Omega(k^{1/3})$, so we may apply Theorem 4.4 to obtain the desired lower bound.

As above, the proof of Theorem 4.5 also establishes a lower bound of $\Omega(m^{1/3})$ on the competitive ratio of the optimal assignment protocol in the multicast model, and we will later

see a matching upper bound. However, as before, this graph gives us only a very weak lower bound, $\Omega(\log n/\log\log n)$, in terms of n. For a polynomial lower bound in terms of n, we may use the same construction as was used in Theorem 4.3.

THEOREM 4.6. *There exists a graph G such that the competitive ratio of any oblivious assignment protocol for G in the multicast model is $\Omega(n^{0.069})$.*

Proof. The graph G is defined by the same construction as in the proof of Theorem 4.3, but this time we choose b by rounding off $(4/3) - (2/3)\cdot\log_2(3) = 0.27669\ldots$ to the nearest multiple of $1/d$. (Note that this value of b still satisfies $3\cdot 2^{b-1} < 1$.) We have already proved that the edge set of G may be partitioned into γ-focal matchings of size $k = |Z|$. Here $m = 2^d$ and $|Z| \geq 2^d - \left(3\cdot 2^{b-1}\right)^d = (1-o(1))m$, so $k = \Omega(m)$ as required by Theorem 4.4. Recall that for this graph G, $\gamma = (1-o(1))\left(2^{2-b}/3\right)^d$ and $d_L = 2^{bd}$. We have chosen b so that $2^{b/2} = \left(1+O\left(\frac{1}{d}\right)\right)2^{2-b}/3$, so $\sqrt{d_L}$ and γ are equal up to constant factors, and the competitive ratio of any oblivious assignment protocol is $\Omega(\sqrt{d_L}) = \Omega\left(2^{bd/2}\right)$. Recalling that $n = 2^{2d}$, this means the competitive ratio is $\Omega\left(n^{b/4}\right) = \Omega(n^{0.069})$.

5 Algorithm for the unicast model

In this section we present an algorithm which is $O(k^{1/2})$-competitive, where k denotes the maximum throughput achievable for the given demand pattern. We will initially work in the fractional-assignment model. Later we will show that a simple randomized rounding of the fractional assignment yields an integral assignment with the same expected throughput, up to a constant factor.

THEOREM 5.1. *There exists an oblivious algorithm which is $O(k^{1/2})$-competitive with the optimum fractional assignment, for every demand pattern D.*

Proof. The oblivious fractional assignment algorithm is extremely simple. Each active client i sends a flow $\frac{1}{d(i)}$ into each of its outgoing edges; each inactive client sends zero flow.

For a server j, recall that the load $\ell(j)$ is defined as the sum of the flows on all incoming edges. Define the flow according to the algorithm specified above, and let Φ be the set of full servers, i.e. those with $\ell(j) \geq 1$. Let $\phi = |\Phi|$. We consider two cases. If $\phi \geq \sqrt{k}$, then one can easily see that the throughput is at least $\phi \geq \sqrt{k}$. Now consider the case in which $\phi < \sqrt{k}$.

First we consider all active clients i with $\Gamma(i) \subseteq \Phi$. This set of clients can send at most ϕ units of flow in the optimum solution, and our algorithm also achieves a throughput of ϕ from the servers in Φ. Now consider a client i which has a neighbor in $j \in V_R \setminus \Phi$. If $d(i) \leq \sqrt{k}$, the flow from i to j is at least $\frac{1}{d(i)} \geq \frac{1}{\sqrt{k}}$. Since $j \notin \Phi$, every unit of flow coming into j contributes to the throughput of our algorithm; in particular, i contributes at least $\frac{1}{\sqrt{k}}$ to the throughput. Now we consider the case in which $d(i) > \sqrt{k}$. In this case, i sends flow $\geq 1 - \frac{\phi}{d(i)}$ to servers not in Φ. Since $\phi < \sqrt{k}$ and $d(i) > \sqrt{k}$, we have $\frac{\phi}{d(i)} < 1 - \frac{1}{\sqrt{k}}$; hence i contributes at least $\frac{1}{\sqrt{k}}$ to the throughput. In both cases, each active client i with at least one neighbor in $V_R \setminus \Phi$ contributes at least $\frac{1}{\sqrt{k}}$ units of throughput. There is no double-counting of throughput in this argument, so we have obtained the desired $\sqrt{k}$ competitive ratio.

Theorem 4.2 demonstrates that no algorithm can achieve a better competitive ratio in terms of k than our simple algorithm, up to constant factors. An obvious corollary of Theorem 5.1 is that our algorithm's competitive ratio, in terms of n, is $O(\sqrt{n})$. This bound is tight in terms of n *for our algorithm*, i.e. there exist instances for which the algorithm's throughput is $O(k/\sqrt{n})$.[1] We do not know if there exists an algorithm achieving a better competitive ratio in terms of n; the best known lower bound is the one specified in Theorem 4.3.

5.1 Rounding fractional to integral assignments

We wish to demonstrate that for any oblivious fractional assignment algorithm A achieving competitive ratio R, there is a randomized integral assignment algorithm A' achieving competitive ratio $O(R)$. If f is the fractional assignment computed by A for a given demand pattern, let A' select a random integral assignment as follows: each active client i chooses a random outgoing edge (independently of the other clients' random choices), with $f(e)$ representing the probability of choosing edge e.

LEMMA 5.1. *Let $\theta(A), \theta(A')$ denote the throughput of A, A', respectively, on the given demand pattern. Then $\mathbf{E}(\theta(A')) \geq \left(1-\frac{1}{e}\right)\theta(A)$.*

The proof uses standard techniques from randomized rounding, and is deferred to the full version of this paper.

COROLLARY 5.1. *There exists a randomized oblivious integral assignment algorithm which is $O(k^{1/2})$-competitive in expectation with the optimum assignment (i.e., maximum matching) for every demand pattern.*

6 Algorithm for the multicast model

In this section, we describe a simple algorithm which achieves a competitive ratio of $k^{1/3}$ the multicast model, where clients are allowed to send their request to more than one server, and a server may select any one of the requests it received and satisfy this request. The algorithm requires no shared random bits, nor does it require the parties to know the structure of the graph G. The clients need only know which servers are adjacent to them, and the servers need only know the degrees of the adjacent active clients. (If necessary, the active clients may communicate this information in their request headers.)

[1] Consider sets A, B, and C, where $|A| = n$, $|B| = n$, and $|C| = \sqrt{n}$. Let $V_L = A$ and $V_R = B \cup C$. The edge set of graph G consists of a perfect matching joining A to B and a complete bipartite subgraph joining A to C. In this example each client has degree at least $\sqrt{n}$. Now, if the adversary chooses A as the set of active clients, then the optimum throughput, k, is equal to n, while our algorithm's throughput is only $O(\sqrt{n})$.

THEOREM 6.1. *There exists an oblivious assignment protocol in the multicast model which is $O(k^{1/3})$-competitive with the optimum assignment (i.e., maximum matching) for every demand pattern.*

Proof. The algorithm is as follows. Each client broadcasts its request to all adjacent servers. If i is a client whose degree in the bipartite graph is $d(i)$, then a server receiving a request from i assigns weight $\frac{1}{d(i)}$ to this request. After receiving all requests, a server chooses to satisfy a random request with probability proportional to its weight.

For a server j, define its weight $w(j)$ to be the sum of the weights of all requests it received. Choose a specific maximum matching M of size k. For every edge $e = (i, j)$ in the matching, at least one of the following must hold:

1. $d(i)w(j) \leq k^{1/3}$
2. $w(j) > k^{-1/3}$
3. $d(i) > k^{2/3}$.

Thus one of the three possibilities is applicable to at least $k/3$ of the edges in M. We deal with them case-by-case.

In case 1, for each matching edge $e = (i, j)$ satisfying (1), the probability that j selects the request from i is

$$(1/d(i))/w(j) = 1/(d(i)w(j)) \geq k^{-1/3}.$$

There are $k/3$ such edges, each has at least a $k^{-1/3}$ chance of being satisfied, and each of them corresponds to a distinct client. Hence the expected number of satisfied clients is $\Omega(k^{2/3})$ as desired.

In case 2, let S denote the set of servers which are right endpoints of matching edges satisfying (2). By assumption, there are $\Omega(k)$ such servers. The fact that they satisfy $\Omega(k^{2/3})$ distinct requests, in expectation, is a consequence of the following lemma which we also use for case 3. (See the proof of Lemma 6.1 in the appendix.)

LEMMA 6.1. *For any real number $0 < r \leq 1$, let S denote the set of servers of weight $\geq r$. The expected number of distinct requests satisfied by the servers in S is at least $\frac{r}{e}|S|$.*

Proof. For each server j in S, flip an independent coin and color it red with probability r. Let $E1$ denote the event that j is colored red, and $E2$ the event that the client i whose request was satisfied by j did not have its request satisfied by any red server other than j. It is clear that $E1$ and $E2$ are independent ($E1$ depends only on j's choice of color, $E2$ depends only on j's choice of job and on the random choices made by other servers.) The probability of $E1$ is r. We claim that the probability of $E2$ is at least $1/e$. To see this, let $d = d(i)$. For each element $j' \in S \setminus \{j\}$ adjacent to i, the probability that j' satisfied i's request is at most $\frac{1}{d(i)r}$, and the probability that it was colored red is r, so there is at most a $1/d(i)$ chance that j' was colored red *and* satisfied i's request. Thus the probability that j' is not a red server satisfying i's request is $\geq 1 - 1/d(i)$. Multiplying at most $d(i) - 1$ such terms together, we get a probability which is at least $1/e$.

Thus the expected number of elements of S satisfying $E1$ and $E2$ is at least $(r/e)|S|$. No client can be satisfied by more than one such server, so altogether the expected number of distinct clients satisfied by S is at least $(r/e)|S|$.

Finally, we address case 3. Partition the servers into two sets, A and B. Set A consists of all servers whose weight is at least 1, all others belong to B. Let X denote the set of clients i such that i is the left endpoint of an edge in the matching M, and $d(i) \geq k^{2/3}$. By hypothesis, $|X|$ is at least $k/3$. For each server j, let $w'(j)$ denote the total weight of the requests it receives from elements of X. The sum of $w'(j)$ over all servers j is simply $|X|$ (since each client contributes exactly one unit of weight, in total), hence one of the following sub-cases applies:

3.1: $\sum_{j \in A} w'(j) \geq |X|/2 \geq k/6$.

3.2: $\sum_{j \in B} w'(j) \geq |X|/2 \geq k/6$.

We handle the two sub-cases separately. For case 3.1, note that $w'(j)$ is bounded above by $k^{1/3}$, because j is adjacent to at most k elements of X, and each of them contributes at most $k^{-2/3}$ units of weight to $w'(j)$. So in order for (3.1) to hold, it must be the case that $|A| \geq k^{2/3}/6$. Applying the lemma above with $r = 1$, we find that the expected number of distinct clients satisfied by servers in A is $\Omega(k^{2/3})$ as desired. For case 3.2, at least $3/4$ of the clients in X have at least $1/3$ of their neighbors in B. (Otherwise these clients would contribute less than $|X|/4$ to the sum on the left side of (3.2), and the remaining $|X|/4$ clients would contribute at most $|X|/4$ to that sum.) For a client with $1/3$ of its neighbors in B, the probability of its request being satisfied is bounded below by a constant, namely $1 - e^{-1/3}$. To see this, let i be such a client and j any neighbor of i in B. The probability that j satisfies i's request is $\frac{1}{d(i)w(j)} \geq \frac{1}{d(i)}$, so the probability that j does not satisfy i's job is $\leq 1 - 1/d(i)$. Multiplying at least $d(i)/3$ such terms together, we get a failure probability which is less than $e^{-1/3}$. So, in case 3.2, we find that the expected number of elements of X whose request is satisfied by an element of B is at least $(1 - e^{-1/3}) \cdot (3/4) \cdot |X| = \Omega(k)$ which easily beats the required $\Omega(k^{2/3})$ bound.

Theorem 4.5 demonstrates that no algorithm can achieve a better competitive ratio in terms of k than our algorithm, up to constant factors. An obvious corollary of Theorem 6.1 is that our algorithm's competitive ratio, in terms of n, is $O(n^{1/3})$. This bound is tight in terms of n *for our algorithm*, i.e. there exist instances for which the algorithm's throughput is $O(k/n^{1/3})$. [2] We do not know if there exists an algorithm achieving a better competitive ratio in terms of n; the best known lower bound is the one specified in Theorem 4.6.

7 Restricted adversary model

Returning from the setting of one-shot (oblivious) algorithms to the online setting, we consider online fractional assignment algorithms for a sequence of demand patterns $D_t : V_L \rightarrow \{0, 1\}$, which may be adversarially specified subject to the restriction that when a client becomes active, it remains active for the next r rounds, where r is a positive

[2] Consider a bipartite graph G whose left vertices are partitioned into two sets A, B and whose right vertices are partitioned into two sets C,D, such that $|A| = k$, $|B| = k^{2/3}$, $|C| = k$, $|D| = k^{2/3}$. A and C are joined by a perfect matching, B and C are joined by a complete bipartite graph, A and D are joined by a complete bipartite graph, and there are no edges from B to D. If each client is active, then it is an exercise to check that the algorithm specified above satisfies only $O(k^{2/3}) = O(k/n^{1/3})$ distinct jobs.

integer known to all clients. (As always, we refer to a client i as active at time t if $D_t(i) = 1$, inactive otherwise.) We do not assume that any of the parties know the structure of the graph G; the only requirement is that clients should know the set of adjacent servers, and they should have common knowledge of a number Δ which is an upper bound on the degree of any client. (Such an upper bound is easy to obtain. For example, if the number of servers m is common knowledge, this is a suitable value for Δ.)

Unlike previous sections, which assumed each server has unit capacity, we assume here that each server j has a non-negative capacity c_j. No upper bound on c_j is assumed, but the capacities are assumed to remain constant over time. The throughput of an assignment is defined to be the sum, over all servers j, of $\min\{\ell(j), c_j\}$, where $\ell(j)$ as always denotes the load on server j.

Our algorithm runs in a series of synchronous, concurrent rounds. In each round, each client assigns load fractionally among the adjacent servers. (As in Lemma 5.1, such a fractional assignment may be converted into an integral assignment by randomized rounding, decreasing the expected throughput by only a constant factor.) Each server sums the assigned loads and reports its load/capacity ratio back to the adjacent clients. This is the only communication in either direction.

7.1 Algorithm The algorithm divides time into windows of length $\lceil r/2 \rceil$. Each active client maintains a fractional assignment of load on its outgoing edges. When a client of degree d becomes active, it waits for the start of the next window and then initializes its fractional assignment by sending $1/d$ units of flow on each outgoing edge. While a client remains active, it updates its fractional assignment f at the end of each round, using the feedback from the adjacent servers as follows. Let $\alpha = (2\Delta)^{6/r}$. A server is defined to be "undersupplied," "comfortable," or "oversupplied," according to whether the corresponding server's load/capacity ratio is $< 1/\alpha$, is in the interval $[1/\alpha, 1]$, or is > 1, respectively. We will refer to edges as undersupplied, comfortable, or oversupplied according to the status of the corresponding server, and for a client i we will denote the total flow on undersupplied, comfortable, and oversupplied edges by $f_u(i), f_c(i), f_o(i)$, respectively. A client i with $d_o(i)$ oversupplied outgoing edges is called "unhappy" if

$$0 < (\alpha - 1) f_u(i) < f_o(i) - d_o(i)/2\Delta,$$

otherwise "happy". A happy client retains the same flow distribution in the next round. An unhappy client redistributes flow from the oversupplied edges to the undersupplied ones, so as to multiply the amount of flow on each undersupplied edge by α. In doing so, the flow on each oversupplied edge may not drop below $\frac{1}{2\Delta}$. (The condition defining an unhappy client ensures that such a redistribution is possible.)

7.2 Analysis In a time window W, call a client *eligible* if it is active in every round belonging to W. Define a modified sequence of demands $\hat{D}_t(i)$ by specifying that $\hat{D}_t(i) = 1$ if i is eligible in the window containing round t, 0 otherwise. The analysis of the algorithm depends on proving that it is $O(\alpha)$-competitive with the throughput of the optimum sequence of assignments for the *modified* demands. The following lemma explains why this is sufficient.

LEMMA 7.1. *Let $\theta, \hat{\theta}$ denote the throughput of the optimum sequence of assignments for the original demands and the modified demands, respectively. Then $\hat{\theta} \geq \theta/3$.*

Proof. Let $f_1, f_2, \ldots, f_T$ be a throughput-maximizing sequence of assignments for the original demands D_t. We may assume that each f_t assigns to server j a load $\ell_t(j)$ which is at most c_j. (If not, we may adjust f_t by reducing the fbw on some of the incoming edges to server j without reducing the throughput.) Now construct a sequence of assignments $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_T$ as follows. Initially, $\hat{f}_t = f_t/3$. For each client i which is active but not eligible at time t, it must be the case that either:

- i became active during the window W containing t. If so, i is eligible in the next window, $W + 1$. Let $t' = t + \lceil r/2 \rceil$.
- i ceased to be active during W. If so, i is eligible in the preceding window, $W - 1$. Let $t' = t - \lceil r/2 \rceil$.

Now adjust $\hat{f}$ by changing $\hat{f}_{t'}(e)$ to $\hat{f}_t(e) + \hat{f}_{t'}(e)$ for each outgoing edge e from i, and setting $\hat{f}_t(e)$ to zero. In this way, we obtain a sequence of assignments $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_T$ such that:

- The outfbw from ineligible clients is zero in each round.
- The outfbw from an eligible client i is at most 1. (In the original assignments f_t, the outfbw from i was at most 1 in each round. In $\hat{f}_t$, the outfbw from i at time t is bounded above by the average outfbw in rounds $t, t - \lceil r/2 \rceil, t + \lceil r/2 \rceil$ of the original assignment.)
- The infbw to a server j is at most c_j. (In the original assignments f_t, the infbw to j was at most c_j in each round. In $\hat{f}_t$, the infbw to j at time t is bounded above by the average infbw in rounds $t, t - \lceil r/2 \rceil, t + \lceil r/2 \rceil$ of the original assignment.)
- The throughput is $\theta/3$. (We initialized $\hat{f}_t$ to $f_t/3$, and we subsequently shifted fbw without changing the combined throughput.)

By definition, the throughput of $\hat{f}_1, \ldots, \hat{f}_T$ is at most $\hat{\theta}$. This establishes that $\hat{\theta} \geq \theta/3$.

THEOREM 7.1. *The algorithm specified in Section 7.1 is $O(\Delta^{6/r})$-competitive.*

Proof. For a time window W, let $\hat{\theta}(W)$ be the optimum throughput achievable by an assignment of the eligible clients only. By the preceding lemma, we know that it is sufficient to prove that the algorithm's throughput during W is $\Omega(\hat{\theta}(W)/\alpha)$. For the remainder of the analysis, we will limit our attention to the time rounds which belong to W.

First, we note that the load on a server cannot increase by a factor of more than α in any round, because the load on each edge cannot increase by a factor of more than α. If a server is comfortable, the load on its incoming edges does not change at all.

Therefore a server may not become oversupplied in the next round unless it was already oversupplied in the current round.

Second, we note that for an edge $e = (i, j)$, the load $f(e)$ does not increase while j is oversupplied; if j ever ceases to be oversupplied, in each subsequent round $f(e)$ either increases by a factor of α or remains the same. Moreover, the number of rounds in which $f(e)$ increases is at most $r/6$ because $\alpha^{r/6} = 2\Delta$, and $f(e)$ is never less than $\frac{1}{2\Delta}$.

For each edge $e = (i, j)$ in each round t, one of the following applies:

1. i was happy in round t.
2. j was not undersupplied in round t.
3. The load on e increased by a factor of α at the end of round t.

We have already argued that the third case applies to at most $r/6$ of the $\lceil r/2 \rceil$ rounds in W. Therefore, at least $r/6$ of the rounds satisfy either the first or the second case.

Call a client "satisfied" if it is happy in at least $r/6$ of the rounds in W; let X be the set of all such clients. Call a server "satisfied" if it is oversupplied or comfortable in at least $r/6$ rounds of W; let Y be the set of all such servers. Above, we have proven that every edge has either its left endpoint in X or its right endpoint in Y. Therefore, in the maximum-throughput flow, every unit of flow goes through either a satisfied client or a satisfied server, resulting in the bound

$$\frac{\hat{\theta}(W)}{\lceil r/2 \rceil} \leq |X| + \sum_{j \in Y} c_j. \tag{7.1}$$

However, it follows from the definition of "satisfied" that the algorithm's throughput θ satisfies:

$$\frac{\theta}{r/6} \geq \max \left\{ \frac{1}{2\alpha}|X|,\ \frac{1}{\alpha} \sum_{j \in Y} c_j \right\} \tag{7.2}$$

The lower bound $(1/\alpha) \sum_j c_j$ is immediate from the fact that a server j which is not undersupplied has throughput at least c_j/α. The lower bound $(1/2\alpha)|X|$ may be derived as follows. If a client i is happy in round t we have: $(\alpha - 1) f_u(i) \geq f_o(i) - \frac{1}{2}$, whence,

$$\begin{aligned} \alpha f_u(i) + \alpha f_c(i) &\geq (\alpha - 1) f_u(i) + f_u(i) + f_c(i) \\ &\geq (f_o(i) + f_u(i) + f_c(i)) - \frac{1}{2} = \frac{1}{2}. \end{aligned}$$

Every unit of flow which i sends to an undersupplied or comfortable server contributes to the throughput in round t. Therefore a happy client contributes at least $f_u(i) + f_c(i) \geq \frac{1}{2\alpha}$ units of throughput in round t, which justifies (7.2).

Finally, putting together (7.1), (7.2), we obtain:

$$18\alpha \left(\frac{\lceil r/2 \rceil}{r} \right) \theta \geq \hat{\theta}(W).$$

References

[1] 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. ACM, 44 (1997), pp. 486–504.

[2] B. AWERBUCH AND Y. AZAR, *Local optimization of global objectives: Competitive distributed deadlock resolution and resource allocation*, in Proc. 35th IEEE Symposium on Foundations of Computer Science, 1994, pp. 240–249.

[3] B. AWERBUCH, Y. AZAR, AND S. PLOTKIN, *Throughput competitive on-line routing*, in Proc. 34th IEEE Symposium on Foundations of Computer Science, 1993, pp. 32–40.

[4] B. AWERBUCH AND T. LEIGHTON, *A simple local-control approximation algorithm for multicommodity flow*, in Proceedings of the 34th Annual Symposium on Foundations of Computer Science, 1993, pp. 459–468.

[5] ———, *Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks*, in Proc. 26th ACM Symposium on Theory of Computing, 1994, pp. 487–496.

[6] Y. AZAR, E. COHEN, A. FIAT, H. KAPLAN, AND H. RACKE, *Optimal oblivious routing in polynomial time*, in Proc. 35th Symposium on Theory of Computing, 2003, pp. 383–388.

[7] Y. BARTAL, J. W. BYERS, AND D. RAZ, *Global optimization using local information with applications to flow control*, in Proc. 38th IEEE Symposium on Foundations of Computer Science (FOCS '97), 1997, p. 303.

[8] M. BIENKOWSKI, M. KORZENIOWSKI, AND H. RÄCKE, *A practical algorithm for constructing oblivious routing schemes*, in Proc. 15th ACM Symposium on Parallel Algorithms and Architectures, 2003, pp. 24–33.

[9] E. COHEN, *Approximate max flow on small depth networks*, in Proc. 33rd IEEE Symposium on Foundations of Computer Science, 1992, pp. 648–658.

[10] N. GARG AND J. KOENEMANN, *Faster and simpler algorithms for multicommodity flow and other fractional packing problems.*, in Proceedings of the 39th Annual Symposium on Foundations of Computer Science, 1998, p. 300.

[11] N. GARG AND N. E. YOUNG, *On-line end-to-end congestion control*, in Proc. 43rd IEEE Symposium on Foundations of Computer Science, 2002, pp. 303–312.

[12] A. V. GOLDBERG AND R. E. TARJAN, *A new approach to the maximum-flow problem*, J. ACM, 35 (1988), pp. 921–940.

[13] M. HAJIAGHAYI, R. KLEINBERG, T. LEIGHTON, AND H. RÄCKE, *Oblivious routing on node-capacitated and directed graphs*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, 2005.

[14] C. HARRELSON, K. HILDRUM, AND S. RAO, *A polynomial-time tree decomposition to minimize congestion*, in Proc. 15th ACM Symposium on Parallel Algorithms and Architectures, 2003, pp. 34–43.

[15] R. M. KARP, E. UPFAL, AND A. WIGDERSON, *Constructing a perfect matching is in Random NC*, Combinatorica, 6 (1986), pp. 35–48.

[16] M. LUBY AND N. NISAN, *A parallel approximation algorithm for positive linear programming*, in Proc. 25th ACM Symposium on Theory of Computing, 1993, pp. 448–457.

[17] C. H. PAPADIMITRIOU AND M. YANNAKAKIS, *Linear programming without the matrix*, in Proc. 25th ACM Symposium on Theory of Computing, 1993, pp. 121–129.

[18] S. A. PLOTKIN, D. B. SHMOYS, AND É. TARDOS, *Fast approximation algorithms for fractional packing and covering problems*, Math. Oper. Res., 20 (1995), pp. 257–301.

[19] H. RÄCKE, *Minimizing congestion in general networks*, in Proceedings of the 43rd Symposium on Foundations of Computer Science, 2002, pp. 43–52.

[20] H. RÄCKE AND A. ROSEN, *Distributed online call control in general networks*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, 2005.